AUTOMATED LANGUAGE ANALYSIS

1968 - 1969

Report on research for the period
March 1, 1968 - February 28, 1969

Sally Y. Sedelow, Principal Investigator
Departments of English and Computer & Information Science
University of North Carolina
at
Chapel Hill

The research reported herein was conducted under Contract N00014-67-A-0321, Office of Naval Research, U.S. Navy, Task No. NR 348-005

Distribution of this Document is Unlimited

The research reported herein was conducted under Contract N0014-67-A-0321, Office of Naval Research, U.S. Navy, Task No. NR 348-005

at the

University of North Carolina
at Chapel Hill

AUTOMATED LANGUAGE ANALYSIS

1968 - 1969

Report on research for the period
March 1, 1968 - February 28, 1969

Sally Y. Sedelow, Principal Investigator
Thomas Gerig, Consultant
Walter A. Sedelow, Jr., Consultant
Walter L. Smith, Consultant
H. William Buttlemann
William G. Hickok
Joan Peters
Larry Rosen
John B. Smith

The views, conslusions, or recommendations expressed in this document do not necessarily reflect the official views or policies of agencies of the United States Government.

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Automated Language Analysis

1968 - 1969

Report on Research for the Period
March 1, 1968 - February 28, 1969

Sally Yeates Sedelow, Principal Investigator

ABSTRACT

This report describes research directed toward unequivocally characterizing the language usage of any writer or speaker. During the past year, a PREFIX program has increased the root-grouping capability of the VIA package, which is a set of programs designed for content, or thematic, analysis of a written or spoken language unit. A ring-structure version of VIA which provides great search flexibility as well as the potential for extended explorations of semantic relationships, as revealed by interconnecting rings, has been further developed. Research on the nature of thesauri has continued and Roget's International Thesaurus has been keypunched to facilitate computer-aided research on its structure. A set of programs designed to show co-occurrence patterns has been implemented, as have procedures for producing non-verbal representations of language-usage patterns.

PREFACE

This project has received support from many sources during the past year. The members of the Information Systems Staff of the Office of Naval Research have been unfailingly helpful whenever advice or administrative support was sought. The University of North Carolina at Chapel Hill has also provided administrative assistance. In addition, the Department of Computer and Information Science has provided office space and equipment as well as secretarial support for the research group and the Department of English has granted the principal investigator a reduced teaching load, freeing time which is partially allocated to this project. We also wish to extend our thanks to all the individuals who have contributed, in one way or another, to the furtherance of this research effort.

TABLE OF CONTENTS

I. Survey of the Computer-Aided Language Analysis Project
March 1, 1968 - February 28, 1969

A. INTRODUCTION

The long-term goal of this research project is to build a system of programs which will provide a comprehensive description of the language usage of an individual or a group of individuals, as well as of larger social units such as a region, or country. It has become increasingly clear that language not only functions as a descriptor of some reality but that it, itself, comprises a reality of major import. That is, individuals and nations respond to the language of other individuals and nations with verbal <u>and</u> non-verbal actions which evoke other verbal and non-verbal actions. Prior to the availability of the computer, research on individual, group, regional, and national interaction tended to concentrate upon non-verbal variables, in part because these are more easily defined and described. Some efforts at non-computer-aided content analysis, such as those undertaken by political scientists and by clinical psychologists, have indicated the value of examining language behavior, as well as indicating the potential labor entailed by any effort to do so thoroughly. Attempts to use the computer for such tasks have shown, of course, that describing language so that it can be dealt with by the computer is an enormous undertaking because human understanding of language usage is not very precise. To document the scale of the task, many research efforts and results could be cited. One statistic from a recent computer run for our own project is probably sufficiently impressive: to show the interconnections, using interlocking rings, between words in the first chapter of the Praeger* translation of Soviet Military Strategy which have some semantic relationship with the word agression, 600 pages of essentially non-redundant computer output were produced; more output would have resulted had not a limit of 600 pages been imposed. Results such as this suggest why language is so little understood. As the Pierce Committee report has urged, basic and comprehensive research on language is vital so that this central component of human life may be intelligently analyzed.

Not only is word choice important, but so are other parameters pertaining to the relative positioning of words. Certain categories, such as syntactic categories, have traditionally been used to describe some aspects of positioning; other categories relating to sound, meaning, or even the

*V.D. Sokolovsky, Soviet Military Strategy, trans. Translation Services Branch, Foreign Technology Division, Wright-Patterson AFB, Frederick A. Praeger, Inc. 1963.

physical appearance of graphemes have been less well explored for their possible utility in describing language usage styles. Differing styles (we define style to include word choice as well as other parameters) can lead not only to sometimes disastrous misunderstandings, but can also serve as clues to shifts from one writer or speaker to another, or from one attitude or point of view to another by a single writer or speaker. The perception of such shifts can be a significant clue to an alteration in attitude or emphasis which may indicate to the perceiver of the shift a desired mode of response.

While the goal of this project is to develop as comprehensive a description of any given style as possible, it has been our aim to design modular programs which are in themselves useful for language analysis. Thus, the VIA package can be used separately from MAPTEXT, and within VIA subprograms can be used to produce indices, to group words together by root, and to perform word-keyed searches, as well as, when taken together, to perform content analysis on a specific subsection or sections of text, using groups formed by words having a common root. MAPTEXT can be used in order to represent any linguistic element or elements specified by the researcher or it can be used to map the output words produced by VIA. As we specify and program given tasks, we try to define interfaces among the tasks so that, when possible, decision-making procedures can be built into the system enabling it to respond dynamically as the characteristics of a given text begin to emerge. This "self-adaptive" capability is a goal, not a fully accomplished reality; but it does serve as a guide for the project as a whole.

B. WORK DURING THE PAST YEAR - SEDELOW RESEARCH REPORT

During the past year, our efforts have been concentrated upon research related to the VIA package and research related to mapping verbal text onto other representations, including statistical representations. In order to increase the capacity of the procedure in the VIA package that makes a first pass at semantic grouping by pulling together words with a common root, a program which deals with prefixes has been written so as to complement the suffix procedure already programmed. The ring-structure version of the text-specific thesaurus building procedure in VIA, which was in a preliminary state at the time of last year's research report, has been considerably developed and run on some large data bases. A start toward providing interactive capabilities for this procedure has been made. The effort to eliminate the manual search now necessary between the root-grouping procedure and the text-specific thesaurus building procedure has resulted in signing a contract with the Thomas Y. Crowell Company permitting research on the Third Edition of *Roget's International Thesaurus*. This step was taken after earlier comparative research* on Webster's Dictionary of Synonyms and two versions of Roget's thesaurus. The International Thesaurus has not been keypunched** and research on the thesaurus is ready to begin.

Research and programming related to mapping the text onto nonverbal representations has moved in graphic and statistical directions. The MAPTEXT program originally written for the Philco 2000*** had not been redone for the IBM 360 system currently used by the project; it has now been rewritten, using PL/I. Standard statistical measures and procedures have been investigated and, in some cases, tested for their value to this project. Among other efforts, work on frequency distributions will continue in an attempt to provide a statistical algorithm for specifying the threshold which determines search keys in the VIA program. The effort to find appropriate statistical procedures for the analysis of graphemic strings, including punctuation marks and blanks, continues. Where possible, already written computer programs are used for the work on statistical parameters of language.

*See Sally Yeates Sedelow, Stylistic Analysis, Report on the Third Year of Research, 1 March 1967, DDD # AD 651-591 and Sally Y. Sedelow, et. al., Automated Language Analysis, 1967-1968, DDC # AD 666-587.

**The keypunching conventions are shown in figure 2 in page 35 of this report.

***See pp. 86 - 113, Sally Yeates Sedelow, Stylistic Analysis: Report on the Second Year of Research, DDC # AD 629-789 and pp. 70 -89, Sally Yeates Sedelow, Stylistic Analysis, Report on the Third Year of Research, DDC # AD 651-591.

1. Recognizing Roots in Words Which Have Prefixes*

On pp. 82-91 of this report, John Smith describes the program he has designed and written for the recognition of prefixes. This program will be available for use, if desired, as part of the word root-grouping section of VIA. From its beginning, VIA has recognized words with common roots but different *suffixes*. The function of suffixes is heavily syntactic; thus, the effective removal of a suffix usually does not seriously affect the central meaning of a word. Because VIA is designed to look for ideas, or concepts, or themes, central meanings (such as *mad* in *mad*ly or *mad*ness) are the appropriate search keys. The function of prefixes is much more heavily semantic, or related to meaning, than is that of the suffixes, thus making the classification of a word's central meaning in terms of its root less reliable.

Linguists and other scholars concerned with language have been interested in affixes because of information they provide about the influence of one language upon another, or about the formation of words in a given language, or about syntactic functions of a word. Recognition of the *semantic* implications of affixation, especially of prefixes, is

*Joan Peters and Will Deland were responsible for consulting other references and lists of prefixes as well as working completely through the *Random House Dictionary* to compile lists of prefixes and exception and inclusion lists.

implicit in the kind of distinction described, for example, by Hockett when he talks of English compounds. Hockett points out that English stems such as telegraph-, telephone-, and so on

> contain at least one constituent which is clearly not itself a stem: *tele-*, *phone-*, *gramo-*. But many contain one constituent which is either a stem or is the same in shape as, and similar in meaning to, some stem: *graph*, *phone*, *photo*, *stat*....
>
> Yet English stems of the kind just dealt with are different from English *phrasal compounds*, like *blackbird*, *bluebird*, *blackboard*. The latter are a special sort of sequence of two words...their structure is syntactical, not morphological. It seems best, for English to...speak simply of *close compounds* (*telegraph* and the like) in contrast to phrasal compounds. The important fact about elements like *tele-*, *phono-*, *photo-*, *graph-*, *phone-*, *gramo-*, *stat-* is that they occur quite freely in close compounds; whether each of them is or is not a stem then assumes secondary importance.*

For the definition of stem used by Hockett, his text may be consulted.** For our purposes it is important to notice that the constituents such as *tele-*, *phone-*, *gramo-* to which Hockett refers may function as prefixes. The problem they pose as functioning prefixes is suggested by the necessity to distinguish between close and phrasal compounds.

*Charles F. Hockett. *A Course in Modern Linguistics*. The Macmillan Company, 1958, p. 243.

***Op. cit.*, pp. 240-244.

As Hockett says, phrasal compounds comprise two words linked together. That is, blue is still blue and bird is still bird in the compound bluebird; hence, "their structure is syntactical, not morphological." In close compounds, however, the meaning of the stem (or root, in our terms) is altered by its prefix--a telegraph is quite different from any form of graph. Should *tele-* be designated as a prefix and *graph* grouped with other words having that root?

The semantics problem does not seem so severe when dealing with prefixes which have the effect of reversing the action or meaning of a word. That is, if one is looking for central ideas or themes, an occurrence of *inimitable* does carry the theme of imitability even though the *in-* prefix emphasizes its impossibility. Many other examples, such as *de*moralization, *de*centralize, *dis*associate, etc., could be cited to support this point.

Sometimes a prefix makes very little difference to the meaning of a root. The notion of *fend*, meaning to keep something or somebody off, is not significantly altered by the addition of *de-*. Nor are *guise* or *rupt* greatly affected by prefixing *dis-*. *Con-* added to *junction* provides another example of a prefix which, in present-day English, is largely redundant.

There are, however, many instances analogous in semantic difficulty to that of the phrasal compound cited above. An inspection of the etymology of the problem prefixes and roots may explain the reasons for their having originally been connected, but those reasons are now generally obscured so far as present usage is concerned. Given current usage, what should be done with the tribute in distribute or the play in display?

There are no adequately tested algorithms for describing the semantic behavior of prefixes. For references relevant to an investigation of prefixes, the bibliography at the end of this section may be consulted. The papers of Resnikoff and Dolby, and Earl represent a promising effort to recognize, automatically, character sequences which may function as affixes. Rules have been formulated to describe these sequences and the recognition of affixes according to these rules is reported to have been quite successful. However, as Resnikoff and Dolby point out, they

> have discussed only the question of determining the affixing strings. The more delicate problem of deciding when an affix is acting as an affix in a particular work remains. For example, the weak prefix RE- acts as an affix in READJUST, but not in READING.*

For the operation of VIA, we need answers to those delicate problems. Since adequate rules are not available, the only approach open to us has been to compile lists of prefixes and of words which should be either included or excluded from the province of a given prefix. Lists available in the references cited in the bibliography (especially Ball, Jespersen, and Marchand) were consulted and the entire Random House Dictionary

*H. L. Resnikoff and J. L. Dolby. "The Nature of Affixing in Written English," Mechanical Translation, Vol. 8, Numbers 3 and 4, p. 89.

was searched to determine which words should be designated as having prefixes and which should not. We decided to err on the side of over-inclusion, permitting the researcher to eliminate incorrectly designated prefixes and hence root forms demarcated by the prefixes. Using the program written by John Smith (pp. 82-91), we are now experimenting with data. The program, itself, runs relatively quickly. Using the PL/I program on the 360/75, 180,000 words were examined for prefixes in nine minutes.

To see how PREFIX affected the grouping of words by root, chapter one of the Praeger translation of *Soviet Military Strategy* was run first through PREFIX and then through SUFFIX. The programs must be run in this order because SUFFIX matches the letters, beginning with the initial letters, in pairs of words until it finds a point of divergence; the assumption is that everything beyond that point in both words *may* be suffixes.

An examination of sections of output picked randomly revealed fifty-six root groups of words which had been affected by the PREFIX run. Of this number, ten had clearly gained useful information and three had clearly been burdened with misleading information. Thirty-nine groups consisted solely of stems remaining after the operation of PREFIX; that is, there happened to be no words in chapter one of *Soviet Military Strategy* which had the same root as any of these thirty-nine groups. Twenty-five of the thirty-nine might provide additional useful information in another textual context and ten of the thirty-nine might provide misleading information in another textual context. Four of the thirty-nine would probably neither help or harm any output. Four of the fifty-six root groups are difficult to classify.

The ten root groups which were helped by the PREFIX run were as follows: 1. able, (en)ables;* 2. capacity, (in) capacitate; 3. courage, (en) couraged; 4. danger, (en) dangering; 5. doubt, (un) doubtedly; 6. integration, (dis) integrating; 7. labor, (e) laboration; 8. moral, morally, (de) moralization, (de) moralizing; 9. sequence, sequences, (con) sequent, (con) sequently; 10. value, (e) valuating, (e) valuation. The group in this list which some might question is number 8. We felt that making the link between moral and demoralizing obvious would be helpful in assessing this particular work. The SUFFIX run will make the presence of roots produced by PREFIX apparent by printing the prefix along with the root. Given this information, the researcher can ignore any roots he deems misleading.

The three groups we felt to be harmed by the PREFIX run were 1. center, centers, (re) cently; 2. tribute, (dis) tribution; and 3. part, partial, (pre) pares (pre) paring. We will certainly alter the lists in PREFIX so that 1 and 3 will not happen again and will probably do so for 2 as well.

The twenty-five groups consisting of just roots produced by PREFIX which might have contributed positively, given another

*In this and all following cases, the prefix for a prefixed word is enclosed in parentheses.

text, were as follows: 1. (en) compasses, (en) compassing; 2. (in) conceivable; 3. (dis) coveries; 4. (in) evitable (evitable is not a likely occurrence, but it is a word); 5. (de) fend, (de) fending; 6. (in) fluence, (in) fluenced, (in) fluences; 7. (en) joyed; 8. (con) junction; 9. (ob) ligations; 10. (out) moded; 11. (ap) ply; 12. (com) promises; 13. (ir) refutable; 14. (en) riched, (en) richment; 15. (dis) rupted, (dis) ruption; 16. (fore) see, (fore) seen; 17. (in) separable; 18. (in) stead; 19. (re) strict, (re) stricted; 20. (un) thinkable; 21. (dis) tinction, (dis) tinctions, (dis) tinguished--these roots could be linked with tincture(s) which provide the reverse meanings taint, affect or having a smattering of knowledge. In the final four cases, the use of PREFIX on texts from earlier centuries might produce positive results: 22. (en) gaged, (en) gagement, (en) gagements--the non-prefixed forms all occurred in the 19th century and earlier and meant e.g., a pledge, or deposit of something of value; 23. (co) ordinate, (co) ordinated, (co) ordinating, (co) ordination--in earlier texts, the non-prefixed forms were used to mean ordered, arranged, etc.; 24. (in) vestigations, (in) vestigates--in earlier texts, vestigate was used the way investigate is now used; 25. (sub) jugate, (sub) jugated--in earlier periods, jugal (yoke) and jugate (joined together) were viable forms.

The ten groups consisting of just roots produced by PREFIX which might have produced misleading information in another text were 1. (per) iod, (per) iods--these odd-looking roots could lead to misgroupings in texts where the word iodine and related forms are used; 2. (per) mitted--the root could be confused with "mitted" meaing to wear mittens; 3. (dis) persal, (dis) persed, (dis) persing--for early texts, the root could be linked to the word "perse," meaing blue, and in some root-grouping procedures, although not ours, the root could be linked to forms of "persecute"; 4. (de)ploy, (de) ployment; 5. (dis) putes--the root could be linked with "put" by our program; 6. (re) sult--the root could be linked with "sult" from (in) sult; 7. (per) taining--in sixteenth century texts, "taining" was a device for catching fish in a river; 8. (dis) tance--the root could be linked with "tan"; 9. (un) til--"til" is a name for a specific plant in the East Indies and another, in Madeira; 10. (pro) vide, (pro) vided, (pro) vides, (pro) viding--the root could be linked with the Latin vide meaning refer to, or see. Some of these possibly harmful linkages are very unlikely to occur. Numbers 7 and 9 would fall in this category. Others, such as numbers 1, 2, and 3 are quite unlikely; 10 is also unlikely unless footnotes are included as part of the text examined by PREFIX. Thus, only four of the ten pose any very serious threat to the efficacy of PREFIX. Any or all of these ten groups can be eliminated from future output by making the appropriate inclusions or exclusions for the lists used by PREFIX.

The four groups which would seem to make neither positive nor negative difference are 1. (per) fection; 2. (un) ite, (un) ited, (un) ity, (un) iversal, (un) iversity; 3. (per) manently; 4. (pro) ve, (pro) ved, (pro) ves. There are no English words to which these roots can be either helpfully or unhelpfully linked. This being the case, PREFIX should be altered so that these roots won't be produced. It should be noted that for all words having prefixes, the original word is retained and given in the output; thus, possible loss of information as well as confusion over roots such as "ve" and "ite" are avoided.

The utility of four of the root groups produced by PREFIX is difficult to assess. These groups are 1. (for) bade; 2. (geo) graphic, (geo) graphical, (geo) graphically; 3. (de) nominator; and 4. (dis) posal, (dis) posed, (dis) position, and (dis) positions. Currently, "bade" is seldom used; when it is, it can mean either to have bidden someone do something or to have bidden someone farewell. "Forbade" would be the antonym of one of these senses but not the other; therefore, sometimes grouping "forbade" with "bade" would be helpful but other times it would not. "Geography" is defined in the Oxford English Dictionary as the science concerned with the description of the earth's surface. Words related to "graphy," such as "graphic," "graphical," and "graphically," are concerned with drawing or writing, or producing by words the effect of a picture. If geographic maps, for example, were being discussed, grouping "geography" and some form of "graph" together would seem desirable. Often, though, such grouping might not be especially helpful. "Nominate" is now rarely used to mean to call by the name of, to designate, or to mention or specify by name. Likewise, "denominator" is seldom used to refer to an individual who denominates or gives a name to something. Most often, "denominator" is used in an arithmetic or algebraic context to refer to the number or variable below the line in a fraction--the divisor. Hence, it would seem that for current texts, the prefix should not be removed; for older texts, perhaps it should be. The "position" in "dispositions" might be somewhat appropriately grouped with "position" (in the sense that disposition implies a new position), but the "posal" in "disposal" or "posed" in "disposed" would seem more questionable additions to such a group. The ambiguities inherent in "disposed" (e.g., to get rid of, to be willing to do something) are rendered more confusing by the "posed" remaining when the "dis" is removed. One can pose a question or adopt a pose; "dispose" does not produce the opposite of either meaning of "pose."

With the possible exception of "denominator," the viability of any of the above groupings would depend upon the senses in which the word or words in the group are being used. For the present, the human researcher will have to decide when he wants to include a root from which the prefix has been "removed" and when he wants to exclude it.

These comments about a small subset of the results produced by PREFIX indicate both the value of any such program and the problems associated with its use. Because any language used for human communication is open to such an enormous range of influences--geography, migration patterns, travel patterns, language-learning patterns, verbal playfulness and inventiveness (as in puns, similes, and metaphors), etc. -- the likelihood of developing manageable algorithms to deal effectively with *all* possibilities, both as to type and date of publication, is near zero. The best that can be done is to discover usage regularities which seem relatively invariant over a wide range of texts; these regularities may then be embodied in a program such as PREFIX with the assumption that modifications can be introduced for texts which are typical. In its present form, PREFIX's positive contribution to text analysis would seem substantially to outweigh its negative inputs. Quite obviously, the entries in the prefix tables can be modified so as to improve its performance further; these modifications will be made. But even though PREFIX can be given finer and finer "tunings," there will, for the foreseeable future, be problems of semantic shifts and complex relationships which will be difficult to resolve. PREFIX and SUFFIX will show the researcher what affixes were involved in any given root group so that the research may, if he chooses, modify the programs' results.

BIBLIOGRAPHY

Ayers, Donald M. *English Words from Latin and Greek Elements.* Tucson: University of Arizona Press, 1965.

Ball, Alice Morton. *Compounding in the English Language.* New York: The H. W. Wilson Company, 1941.

Ball, Alice Morton. *The Compounding and Hyphenation of English Words.* New York: Funk and Wagnalls Company, c. 1951.

Baugh, Albert C. *A History of the English Language*, second edition. Appleton-Century-Crofts, 1957, pp. 76-77, 218-220, 366-367.

Bloomfield, Leonard. *Language.* New York: Holt, Rinehart and Winston, Inc., 1933.

Bradley, Henry. *The Making of English.* New York: The Macmillan Company, 1904, pp. 134, 135-136, 139-141.

Bryant, Margaret M. *Modern English and Its Heritage.* 1962, pp. 71, 245-251.

Earl, Lois L. "Structural Definition of Affixes from Multi-syllable Words," *Mechanical Translation*, Vol. 9, No. 2, 1966, pp. 34-37.

Earl, Lois L. "Part-of-Speech Implications of Affixes," *Mechanical Translation*, Vol. 9, No. 2, 1966, pp. 38-43.

Gibbs, J. W. "English Prefixes Derived from the Greek," *The American Journal of Science and Arts*, series 2, VI (November, 1848), 206-9.

Gleason, H. A. *An Introduction to Descriptive Linguistics.* Holt, Rinehart, and Winston, 1961.

Greenough, James Bradstreet, and Kittredge, George Lyman. *Words and Their Ways in English Speech.* New York: The Macmillan Company, 1901, pp. 187-92.

Haldeman, Samuel Steman. *Affixes in Their Origin and Application, Exhibiting the Etymological Structure of English Words.* Philadelphia, 1865.

A Handbook of Anglo-Saxon Derivatives on the Basis of the Hand-book of Anglo-Saxon Root-Words. By a Literary Association. New York: D. Appleton and Company, 1855, pp. 58-64.

Hart, Archibald. The Latin Key to Better English. New York: E. P. Dutton and Company, Inc., 1942, pp. 21-36.

Harwood, F. W., and Wright, Alison M. "Statistical Study of English Word Formation," Language, XXXII (1956), pp. 260-73.

Hockett, Charles F. A Course in Modern Linguistics. Macmillan, 1958.

Jespersen, Otto. A Modern English Grammar on Historical Principles. Vol. VI. Copenhagen: Ejnar Munksgaard, 1942.

Kennedy, Arthur G. Current English. Boston: Ginn and Company, c. 1935, pp. 335-6, 342-6.

Kruisinga, Etsko. A Handbook of Present-Day English. Part II: English Accidence and Syntax. 3rd ed. Utrecht: Kemink and Zoom, 1922.

Marchand, Hans. The Categories and Types of Present-Day English Word Formation, A Synchronic-Diachronic Approach. University, Alabama: University of Alabama Press, 1966.

McKnight, George H. English Words and Their Background. New York: D. Appleton and Company, 1923, pp. 171-6.

Morris, Richard. Elementary Lessons in Historical English Grammar and Containing Accidence and Word-Formation. London: Macmillan and Company, 1891.

Nesfield, J. C. English Grammer Past and Present. London: Macmillan and Company, Ltd., 1901, pp. 391-408.

Partridge, Eric. Origins: A Short Etymological Dictionary of Modern English. Routledge and Kegan Paul, 1958.

Perry, William. The Synonymous Etymological, and Pronouncing English Dictionary. London, 1805.

Prindle, Lester M. "Some Negative Prefixes in English," Classical Weekly. New York, XLI (1948), 130-3.

The Random House Dictionary of the English Language, ed. Jess Stein. Random House, New York, 1966.

Read, Allen Walker, "English Words with Constituent Elements having Independent Semantic Value," The Malone Anniversary Studies, edited by T. A. Kirby and H. B. Woolf, Baltimore: The Johns Hopkins Press, 1949, 306-12.

Resnikoff, H. L., and Dolby, J. L. "The Nature of Affixing in Written English," *Mechanical Translation*, VIII (1965), 84-9.

Resnikoff, H. L., and Dolby, J. L. "The Nature of Affixing in Written English, Part II," *Mechanical Translation*, IX (1966), 23-33.

Robertson, Stuart. *The Development of Modern English*. 2d. ed. revised by Frederic G. Cassidy. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1954, pp. 194-202.

Sheard, J. A. *The Words We Use*. New York: Frederic A. Praeger, 1962, pp. 51-61.

Smart, Benjamin Humphrey. *Walker Remodelled: A New Critical Pronouncing Dictionary*. London, 1836.

Smith, Logan Pearsall. *The English Language*. New York: Henry Holt and Company, c. 1912, pp. 85-97.

Smith, S. Stephenson. *The Command of Words*. New York: Thomas Y. Crowell Company, c. 1935, pp. 68, 69, 70, 107, 109.

Sweet, Henry. *A Primer of Historical English Grammar*. Oxford: The Clarendon Press, 1893.

Walters, R. G. *Word Studies*. Cincinnati, Ohio: South-Western Publishing Company, c. 1949, pp. 29-36.

Zandvoort, R. W. *A Handbook of English Grammer*. 2d ed. Groningen, Batavia: J. B. Wolters, 1946, pp. 277-325, 329-61.

2. Principal Component Analysis: A Discussion

Principal Component Analysis has both advocates and critics. If the criteria for component selection are ill-defined, the correlations produced by principal component analysis are likely to be meaningless, at best, and misleading, at worst. On the other hand, given well-defined selection criteria, some researchers such as Harway and Iker* have concluded that principal component analysis can provide useful information about the behavior of the designated components. We have used a principal component analysis program on chapter one of the Praeger translation of Soviet Military Strategy in order to compare the kind of information it produces with that provided by the VIA and MAPTEXT programs. The purpose of the comparison was to see whether principal component analysis or some analogous statistical procedure would be a useful addition to our package of programs, whether it might replace some of our programs, or whether it should be discarded as a possibility. The programs necessary to prepare the text for the principal component analysis package program were written by John Smith and are described on pp.103-114 of this report. For those unacquainted with the principal component analysis model, an extended description is provided on pp.92-99.

*See reference on p.93 of this report.

For the run on chapter one of *Soviet Military Strategy*, the "components" consisted of all root groups having a frequency of twenty or more. For example, one group included *accomplish*, which had a frequency of 5, *accomplished*, 7, *accomplishment*, 9, and *accomplishments*, 4. Altogether, there were 102 such groups. The size of the textual units within which co-occurrences were plotted was 100 words. Thirty-eight principal component or factor groups were produced by the program.

Primarily, the principal component analysis revealed two major kinds of patterns: 1. the co-occurrence of two words so often that they could be assumed to be used as a pair; 2. thematic or conceptual patterns within a defined space.

Of the thirty-eight groups, twelve were dominated by word pairs for which the factor loadings were so similar as to indicate that the words very often occurred together. For example, in the first group, *armed* had a loading of - 0.796 and *forces* of - 0.726. In fact, *armed forces* is a frequently recurring phrase in chapter one of *Soviet Military Strategy*. Other such pairs were: *foreign policy*, *high morale*, *socialist countries*, *United States*, *most important*, *means* (to)*accomplish*, *bourgeois theory*, *new problems*, *general principle*, *American interests*, *must plan*. *Soviet government* was another pair which correlated quite well but not as highly as the first twelve listed above. It is important to note that the program, itself, gives no information concerning the *order* of the words in the pair. For the above listing, the order which seems most likely is used; however, it is always possible that the words do not form a pair, even though their loadings are similar, and the order inferred on the hypothesis that they are a pair is therefore in error.

MAPTEXT* is the program in our current package which *does* readily display information about order. In a MAPTEXT representation of chapter one of *Soviet Military Strategy*** the contiguity of *armed forces*, in that order, is clearly shown. Because the MAPTEXT run shown in Appendix E was not geared to the output from this recent principal component analysis run, information about the other pairs does not appear. These pairs *could* be given as input to a MAPTEXT run and their order would be revealed. If the only information desired were just the order of these pairs, a modified MAPTEXT program might display just the relative positions of the word pairs, ignoring the rest of the text. In this latter case, words rather than symbols could be used, if the researcher desired. For more extensive mappings, symbols seem preferable because of the "semantic noise" introduced by words and because of the display space reduction made possible by symbols.

*For a listing and sample output from our first PL/I MAPTEXT program, see Appendix E of this report. MAPTEXT was earlier available in FORTRAN and TAC on the Philco 2000.

**See Appendix E of this report.

One might ask why, if MAPTEXT reveals order, there is any need for the principal component analysis output? One possible reason is that MAPTEXT merely accepts as input the linguistic elements specified by the researcher or by other computer programs; at present, MAPTEXT cannot select the elements it will map. (It might also be noted that the major role initially envisioned for MAPTEXT was not to reveal the order of co-occurring words; this information can be extracted, albeit more laboriously, from our indexing program. MAPTEXT was intended to show the spatial relationships, or dynamics, of word groups such as those produced by VIA across an entire text or large subsection of text. Principal component analysis does not provide that kind of information.)

Despite these differences, so far as the word-pair subset of output produced by principal component analysis is concerned, the most closely related program in our package is MAPTEXT. It can reveal the pairs if they happen to have been included in its input. Of course, MAPTEXT could be given exactly the input which is given to the principal component analysis program--in this case, all root groups occurring twenty or more times. The word pairs or, more properly, root pairs would appear, as would their order. In the mapping of chapter one of Praeger reproduced in part in Appendix E, there are seventeen input groups. The word pairs show up clearly but it is necessary at present to count up by hand the number of occurrences in order to know which words occur together sufficiently often to be considered a frequently recurring pair in a given text. The necessity for such hand counts, of course, could be eliminated by giving MAPTEXT the facility for co-occurrence tabulation and for the determination of a significant threshold. Then MAPTEXT would provide the kind of cross-referencing which MAPTEXT and principal component analysis could now provide jointly. Thus, if principal component analysis does not contribute significantly in some other way, it may be an unnecessary tool for our research, given the concept of a MAPTEXT program. To examine further the possible contribution of principal component analysis, this description will now turn to the other major type of information it provided about chapter one of _Soviet Military Strategy_.

The revelation of words which occur so often together that they can be assumed to form a pair is one kind of information carried by the groups produced by the principal component analysis program. Another kind of information might be described as thematic or conceptual: the words are not so highly correlated as to be paired, but they do occur in the same textual context with relatively high frequency and sometimes a theme or concept can be inferred from the words. For example, in factor 2, the words _nature_, _determine_, _strategy_, and _influence_ are correlated; one inference might be that the text is concerned with determining the nature of strategy--_influence_ would seem related to _determined_. In factor 9, the words _changes_ and _weapons_ are the most highly correlated. The words for some of the other factors are as follows: 11. _capital_, Britain, industry; 12. strategy, operation; 16. development, theory; 17. German, production; 19. accomplish, means, aims; 21. leadership, organization; 24. increase, production; 25. material, requirements; 30. capitalist, system; 31. achieve, success, result; 32. world, American, interests; 33. potential, economic, plans.

The program in our package which has a goal somewhat similar to this use of principal component analysis is really the set of programs collectively called VIA (for Verbally-Indexed Associations). A VIA run (again made at an earlier time than the run using the principal component analysis program) on chapter one of Soviet Military Strategy produced output for a number of words listed in the factors above including strategy, weapons, capital, development, theory, production, means, aims, leadership, organization, increase, production, requirements, world, American, economic, and plans. To illustrate the difference between the two programs, the VIA output for weapons is reproduced below with the exception of some special notation, such as the asterisk, which is not relevant to this discussion.

The output in Figure 1 are typical for a list-structured VIA run on a chapter-length text. Clearly, it is very different from the output for factor 9, in which the two important words are changes and weapons. Major differences between the operating design of the two programs derive from the fact that the principal component analysis program is designed to reveal component co-occurrence which must be defined within subsections

```
WEAPONS
       WEAPON
       ------ARM
             -------DISARM
             -------FIREARM
             -------GUN
             -------MISSILE
             -------NUCLEAR
             -------ROCKET
             -------STEEL
             -------WAR
             -------WEAPON
       ------BOMB
       ------MISSILE
             -------MILITARY
                    --------FIGHTER
                    --------GENERAL
                            -------COMMON
                            -------HABITUAL
                            -------NATURAL
                            -------NORMAL
                            -------REGULAR
                            -------TOTAL
                            -------TYPICAL
                            -------UNIVERSAL
                            -------WHOLE
                    --------GUNMAN
                    --------MAJOR
                    --------MARSHALL
                    --------OFFICER
                    --------SOLDIER
                    --------WARFARE
                    --------WAR
             -------NUCLEAR
             -------ROCKET
             -------WAR
       ------MORTAR
       ------RIFLE
       ------SWORD
```

FIGURE 1

of whatever textual unit is being examined; the VIA program uses a "component," or high-frequency root group, as a key for a search of the entire chapter for all other semantically-related words. Thus, other than the main search unit (the chapter in this case), for VIA co-occurrence is semantically defined, rather than spatially defined as it is for principal component analysis.

The resulting output differences are obvious. VIA takes one root group, which may be said to represent a theme or concept, and shows its elaboration within the given relatively extensive textual unit. The elaboration could be even more extensive; for example, another run of this particular version of VIA on chapter one from Soviet Military Strategy would have resulted in the addition of all the groups of words related to the word war. Even so, the output shown in Figure 1 is quite extensive and it carries the connotation of various forms of war (nuclear and conventional warfare); it also "blunders into" a group (the words connected with general) which is clearly extraneous. As it happens, general most often means common, habitual, natural, etc. in Soviet Military Strategy and the wordsattached to it in the VIA output emphasize that meaning. So that in this instance, the elaboration, which had a potentially misleading element, was self-correcting; this is sometimes, but not always, the case.

Principal component analysis may provide a modest semantic elaboration of a given term. Examples in the list above are 21. leadership, organization and 31. success, result. Another example occurred in factor 6 with the words coalition, alliance. More often, however, the words do not bear any close semantic relationship to each other but represent, instead, the spatial convergence of disparate terms. This convergence is often suggestive of a particular theme or idea (e.g., material requirements in 25 above or capitalist system in 30) but the nature of the inference involves more possibilities (syntax becomes important as well as semantics) and more opportunity for error than is often the case with the groups of more closely semantically-related words in VIA.

Nonetheless, principal component analysis does provide information, namely that concerning co-occurrence within a highly restricted textual space, that VIA does not. We have used VIA profitably upon scenes within a play and upon groups of paragraphs, but 100-word units are too small for the efficient utilization of the elaborate VIA apparatus. Again, then, it is MAPTEXT rather than VIA which emerges as the leading alternate to principal component analysis. MAPTEXT will, of course, show co-occurrence within textual units of any size. However, when the input to MAPTEXT becomes too dense, MAPTEXT becomes difficult to read. A possible sequence, then, would be to use a principal component analysis program, or some other procedure which provides co-occurrence information, to reduce the information to manageable proportions and next feed the results to MAPTEXT for a visual portrayal of the spatial relationships, including order. Another possible combination of such programs would be to run VIA, take the resulting conceptual elaborations (these would be very extensive when using the ring-structure VIA) and use them as input to some statistical data reduction program, and use these results as input to MAPTEXT, as suggested above. It is our assumption that when some statistical procedures, such as principal component analysis, are used in isolation, they do not provide a sufficiency of information for reliable employment; it is likely, though, they will have utility when used in combination with other verbal data organization and display procedures. We mean to explore such possible combinations.

3. Thesaurus Research

As noted earlier in this report*, Roget's International Thesaurus has been keypunched in preparation for research on its structure. The conventions used for keypunching are given in Figure 2.

William Buttelmann has written two processing programs to prepare the thesaurus for examination by the computer. The first of these programs, ROGET, will identify individual entries and categories from the text stream of the thesaurus. The output will be a list of pairs having as format the category number and entry. The second of these programs, RUPDATE, will update the output from ROGET with cards read in through the card reader.

Larry Rosen has been considering using rings for data structure organization to facilitate one approach to delineating the structure of the thesaurus. His proposal follows on page 37.

*See p. 11.

Keypunching Conventions for Roget's Thesaurus

I. General.

One card per line; a line is one line in a column; two columns to the page.
Order the cards by column.
Hyphenation is preserved, as is.

II. Type Fonts (Group Shifts).

	text font		Upshift character
a.	Normal text and universal downshift	-	#
b.	Bold face text	-	@
c.	Italics text	-	$
d.	Headings (all caps)	-	%
e.	Parts of speech (sans serif caps) and xrefs (sans serif numerals)	-	¢

III. Capitalization (Bouncing Shift). - *
(needed only in 11a, 11b, 11c)

IV. Diacritics (key before the letter).

text diacritic	Keyed character
´	| (vertical bar)
`	+
^	=
¨ (umlaut)	_ (underscore)

V. Special characters.

(If a character used as a shift character appears in the text, it is to be keyed twice. E.g., for "*" in the text, key "**"; for "$" in the text, key "$$", etc.)

text character	keyed character
ς	0-8-2 punch
[	<
]	>
∿	¬
___(long hyphen or dash)	--(double hyphen)
¿	0-9-1 punch
g̃	0-9-2 punch
£	0-9-3 punch

Figure 2

A Proposal for Research on Roget's International Thesaurus

by Larry Rosen

The possible organization of Roget's International Thesaurus into ring-structured format in computer accessible form suggests certain experiments useful for studying the relationships and clusterings of words in the thesaurus. For example, we begin by assuming that all words linked to each other, no matter how tenuous or distant the link, are actually related to each other in "concept." All words so related are placed in the same "equivalence class." For the purposes of the first phases of this research, the following "equivalence" relationships will hold:

1. No word can be in more than one equivalence class.
2. No two words can be at all linked to each other and be contained in two separate equivalence classes.
3. No word can be listed more than once in any equivalence class -- thus ignoring the effect of multiple "meanings" for any word.

The immediate objective of the research is to determine, using either a ring-structured or tree-structured version of Roget's International Thesaurus as a data base, the number of equivalence classes in the thesaurus.

For example, consider the word MEMORY. The published index (which will not be used, however, and which has not been keypunched,) to the Thesaurus lists the following classes of words associated in concept with the word MEMORY:

348.	AUTOMATION	--	(via ELECTRONIC BRAIN)
535.	MEMORY	--	(via REMEMBRANCE BY MEMORY FROM MEMORY IN MEMORIAM)
875.	CEREBRATION	--	(via COMMEMORATION)
912.	REPUTE	--	(via FAME)
536.	FORGETFULNESS	--	(via HAVE A SHORT MEMORY)

Not only are the listed words associated (perhaps indirectly) with the word MEMORY, but *every* word listed under the above five headings is so related. *All* of the words are placed in the same equivalence class. Using this example from the published index, the process is continued by looking in the index for all words now in the equivalence class and adding to the class all related words thus found. (In the actual experiment as performed on the computer, all nodes connected to the given node will be searched.) The process terminates when:

1. There are no more links to any words not already included in the equivalence class, *or*
2. There are no more words in the Thesaurus.

The following hypotheses are made concerning expected results:

1. The number of equivalence classes is small, perhaps only one.
2. If there are more equivalence classes than one, it should be possible to determine the "concept" around which each equivalence class is formed. This may provide some clues toward a semantic analysis of the Thesaurus.

By clustering words as described above, we have lost or ignored certain information useful in the analysis of a language: the extent of the relationships between words. To recover that information, we can vary the depth of the search at which links are examined and used; as a result, the number of equivalence classes should vary. At some point, it is hypothesized, the number of equivalence classes will be workable. That is, certain links are too tenuous and may result in the inclusion of words in an equivalence class that do not really belong there. By examining the results for each level of inclusion, some idea of the biases, direct or indirect, in the thesaurus itself, may be determined. At this point, comparison between this thesaurus and other thesauri and word-lists can be useful. A side-effect of this research may be the redesign of the thesaurus to limit the biases which result in spurious and incorrect word linkages and word choices.

The problems associated with this research at the present time concern the mechanics of the programming involved. Some method must be found to recursively include words in an equivalence class without exceeding the storage available on the computer, and without using up an excessive amount of computer time. Once a word is placed in an equivalence class, all references to that word must be removed from the thesaurus itself to prevent its being included in any other equivalence class. A revision of the data-set structure of the thesaurus, described elsewhere, will be necessary to include flags for the deletion, in place, of records that have already been used. These problems have not yet been resolved.

Two utility programs have been written to list the Thesaurus. The first, an IBM supplied utility program, simply concatenates all files containing the Thesaurus into one file, and adds sequence numbers to each record. This will allow for the future addition to, or correction of, the Thesaurus.

The second program is a short PL/I program to read the sequenced Thesaurus and print it. It does no reformatting, and, as presently written, does not allow for the selective printing of records. This capability can easily be added when the need arises.

4. Ring-Structure Version of VIA. (By William Buttelmann)

This section is an updated description of a second version of VIA (Verbally Indexed Associations). The initial version of VIA is described in TM 1908/100/00, Stylistic Analysis: First Annual Report, DDC # AD 613-291, 1 March, 1965, and TM 1908/009/00, Updating of Thesaur Program, 17 December, 1965, DDC # AD 629-368.

This newer version of VIA incorporates two major technological changes in the system structure. First, the thesaurus is organized as a ring-structure,* instead of the tree structure previously used. The ring-structure is more general than the tree and is precisely the structure of

*For the notion of the ring structure, we are indebted to the DEACON Project. See James A. Craig, Susan C. Berezner, Homer C. Carney, and Christopher R. Longyear, "DEACON: Direct English Access and Control," in AFIPS Preceedings (FJCC), 1966, pp. 365-380.

current printed thesauri, whereas the tree structure is an approximation to it. Second, the programs are written to take advantage of the large data file random-access capabilities of third generation computers. This means 1) that they are designed to operate on a very large text, with a very large thesaurus (on the order of 1 million entries) and 2) that the text analyses and thesaurus searches and constructions have been designed with flexibility of searching in mind (e.g., one may use the system to look for content relationships in the text, either with other words in the text or with content categories in the thesaurus, but not in the text; and one may use the system to generate microthesauri specific to a given text). Finally, this system has been built with an interactive time-shared version in view.

All the capabilities of the earlier VIA remain. In addition, this version has the ability always to print the words actually occurring in the text, even though they do not appear in the thesaurus, so that the comment "DIFFERENT FORM APPEARS IN THE TEXT" will never appear in the printout.

The remainder of this section is a general description of the systems programs, in order of processing. This version of VIA is structured in four sections: "Text Segmentation", "Root Matching", "Significance Identification and Thesaurus Construction", and "Search and Print." The programs are written in PL/I, and were developed on the IBM S/360 model 75. Since PL/I is a problem-oriented language which is designed to be machine-independent, presumably the system can be run on any machine whose configuration has a PL/I compiler and a random-access memory extension (such as a disk drum, or bulk core store). The system can also be run on smaller machine configurations, but with a limited thesaurus capacity. (E.g., it will run on the IBM S/360 model 40 with a 256K core memory with a thesaurus of better than 10,000 entries, 1,000 words, and 100 categories.)

4.1. Text Segmentation

This section consists of the program INDEX and a sort. The purpose of this section is to separate the stream of text to be analyzed into significant segments. For consistence, we will call significant segments "words", although they may in fact be words, word groups, phrases, or idioms. INDEX formats the input text into variable length records, each containing one word with index information giving its location in the text. The sort then sorts the words into alphabetical order.

4.2. Root Matching

This section consists of the programs PREFIX and SUFFIX and a sort. Their purpose is to associate all single words having the same root, and to eliminate certain "non-content" words, often called "function words" in the literature. English prefixes and suffixes differ considerably in syntactic complexity and semantic significance: prefixes are in general more semantically significant and have much simpler syntax than suffixes. Accordingly, the PREFIX and SUFFIX programs are designed to function differently. PREFIX recognizes prefixes, by comparison with a standard list, together with an inclusion/exclusion list of stems. The prefix is recognized so that SUFFIX may work properly on the root. After the prefixes are marked, the text is again sorted into alphabetical order. Finally, SUFFIX scans the text and recognizes all words that differ only by a suffix. All such words are assumed to have the same stem (the process is refined by an extensive exclusion list) and are assigned a match count (MATCNT) number for future identification. The suffixes are not deleted. Thus, all words with the same MATCNT number have the same root; all words with the same root (and with no prefix differences) have the same MATCNT number. SUFFIX also does the function word elimination, by comparison with a table of standard function words, and prints the index.

4.3. Significance Indentification and Thesaurus Construction

This section consists of the program THESAUR, a sort, and a special merge program, KEYUP. THESAUR processes analysis requests issued by the user, updates the existing thesaurus with information from the text, and identifies the significant words or categories for which relationships in the text are desired. The key words and key categories - or "search keys" as they are called - are sorted and merged (in the program KEYUP) with the search keys from any previous sections of the text. The merged keys are passes to phase 4, where each is used to generate a thesaurus search.

Analysis requests are processed in the section of THESAUR labelled REQUEST. Analysis requests are entered 1 per card (see Section II.B.2 "ANALYSIS Request Cards" for a complete description of the cards and their parameters). They are edited, batched, and stored on TYPE by the program. The number of requests that may be entered per run is limited to the size of the REQUEST_TABLE, normally set at 100.

Updating the existing thesaurus is done by comparing the text with the VOCAB data set portion of the thesaurus. See the INITIALIZE_VOCAB and UPDATE_VOCAB sections of THESAUR. First, the VOCAB is initialized to remove any spurious information which may have been left by earlier processing. Then a garbage collection is done to ensure that there are no imbedded blank records. Finally, the text is scanned sequentially. Each text word is processed as follows: If the word already appears in the VOCAB, its MATCNT and COUNT are entered in VMATCNT and VCOUNT. If this is the first section of text in which the word has appeared, the current TEXTSECT is inserted in VSECT. If the text word is not in the VOCAB, THESAUR attempts to link it with other words that are in the VOCAB. If it can establish a link, it makes a new entry for the word in VOCAB, marks it as a "temporary entry", and enters the MATCNT and COUNT. A link is said to exist if there is another word in the VOCAB with the same root as the text word in question. Since such words may exist in the VOCAB, but without their VMATCNT entered, THESAUR uses the external subroutine STEM to examine likely candidates, if it cannot find entries with a matching VMATCNT.

Finally, if no link can be established, THESAUR produces the message: "WORD_____. UNABLE TO ESTABLISH ANY RELATIONSHIPS IN THESAURUS."

Each analysis request causes a search of the thesaurus and text information for key words and/or categories. One of the most powerful features of this version of VIA is its flexibility in identifying significant thematic content - in choosing significant keys. They may be designated _a priori_ by the user, or the user may ask THESAUR to pick them from the text on a frequency basis. The TYPE parameter in the analysis request card specifies the method of key identification. TYPEs '3' and '4' are the _a priori_ types: one may designate a single word (which need not be in the text) or a particular category in the thesaurus. TYPEs '1' and '2' allow a means for specifying thematic significance based on frequency of occurrence. TYPE '2' bases significance on the sum of the frequencies of all words with the same root. Every root that occurs more often than a specified threshold is considered significant, and a key is generated for each word with that root. More often, however, thematic sameness is a broader classification than root equivalence: words with different roots may signal the same theme; words with the same root may signal different themes. The relationships depend both on the orientation of the author and the viewpoint of the analyst. One may wish to vary such relationships from analysis to analysis. Thematic similarity is precisely what the thesaurus _categories_ are intended to describe.

TYPE '1' analysis bases significance on the sum of the frequencies of all words in the same category. Every category that occurs more often than the threshold is considered significant, and a key is generated for it. TYPE '2' keys are generated in the UPDATE_VOCAB section of THESAUR. All the others are generated in the BUILD-KEYS section. A complete description of the TYPE parameter is given in Section II.B.2, "ANALYSIS Request Cards."

The obvious advantage of a category-based count over a root-based count is that significant content may be the accumulative effect of the occurrences of several roots, no one of which occurs frequently enough to be detected by the root-based method. The disadvantage is the cost of extra processing time on the computer. (Of course, it is not necessary that the categories in the master thesaurus be formed on the basis of thematic criteria. The categories of the master thesaurus are a priori to any text analysis, and one may choose a thesaurus organized any way he likes.)

An alternate approach to the counting method would be to use frequencies relative to normal usage. This requires tables of words and their frequencies, tabulated from random samples of the language taken from a very large population. Regrettably such tables are not, in general, available.

4.4. Search and Print

This section is the program SPRINT which searches the thesaurus for content-word and word-category relationships keyed by the search keys generated by THESAUR. All the textual information needed to direct the search has been entered into the thesaurus by the data preparation section, so no further references to the text file are needed.

SPRINT consists chiefly of a recursive subprocedure which, cued by a key word, searches through the thesaurus to find all the semantic categories containing the key word. Within each category, it uses each related word to key another level of search. This process occurs through any number of levels of recursion, up to 9. If the key is a category, the search begins by finding all the words in that category. The number of levels is specified in the DEPTH parameter of each analysis card. Because of the cyclic, or "ring" structure, of the thesaurus, certain redundancies are inevitable. For example, every word is related to itself. More intricately, two or more words in several categories together will cause the search to return eventually to the first category and thus repeat itself. Such redundancies are recognized by the program and suppressed.

Considerable flexibility is allowed in the kind of searching that is done. This flexibility comes from the fact that the base thesaurus contains many words not in the text. Some of these, however, are related to words in the text, and it may be important to know them. Thus, for example, it is possible to request a search for all words related to a given key word, even though the key word itself is not in the text at all. Also, for example, it is possible to ask for words in the text that are related in the thesaurus, even if the words that establish the link are not themselves in the text. These options are controlled by the TYPE and MODE parameters specified in the analysis request cards. (See Section II.B.2. for the format of the ANALYSIS Requests.)

The shape of the printed output for a search based on a single key is called a "tree." The key is the "root" and the words at the lowest level of search (the farthest right on the paper) are called "leaves." Each word or category that is used as a branch point for further searching is called a "node." The root and leaves are also nodes. Level or depth of search corresponds to level in the tree: the root is level zero.

There are five search modes one may choose: Text-limited, Text-oriented, Text-rooted, Text-related, and Subthesaurus. These are described completely in the section "ANALYSIS Request Cards" under the MODE parameter.

When VIA is used to analyze a text which has been separated into sections (see documentation of TEXTSECT card in section II .B.l.), it is convenient to know when new words appear in the text. To flag their appearance in the SPRINT printout, new words are preceded by a short dash line. It is also often desirable to conduct searches on keys which have been keys in earlier sections of the text, even though they are not keys in the current section. Such keys are preceded by an asterisk when they are printed as the root of a tree. Words which are in the thesaurus but not in the current section of text may sometimes be printed because they establish links between text words. In *all* cases, words not in the text are printed enclosed in parentheses.

C. PLANS FOR FUTURE RESEARCH AND DEVELOPMENT

During the coming year, research connected with this project will concentrate upon two major areas -- thesaurus structures and ring-structure output problems -- related to VIA, upon increasing facilities for the non-verbal representation text, and upon moving toward some interactive capability.

Some of the reasons for the research on *Roget's International Thesaurus* have been suggested earlier in this report. For the complete automation of VIA, a general-purpose thesaural-like reference in computer-accessible form is needed. A general-purpose reference is desirable because such references represent assessments across time of the usage of a given language. For highly specialized texts or research goals, special-purpose dictionaries might be added to the general purpose base; ideally, the base might be said to comprise an averaging of many people's word association nets. As is well known, no such ideal base exists. Having done comparative VIA runs with *Webster's Dictionary of Synonyms*, *Roget's University Thesaurus* and *Roget's* International Thesaurus,* we decided that the *International*

*See footnote * on page11 of this report.

Thesaurus merited further exploration. We also have computer access to the million-word corpus of American Standard English prepared at Brown University under the direction of W. Nelson Francis and we will have access to the Random House Dictionary of the English Language. We hope to compare these latter texts with the International Thesaurus, both for the purposes of revealing bias and for modifying bias when it is discovered. An investigation of the internal structure of the International Thesaurus, itself, is also planned, perhaps along the lines suggested on pp.37-40 of this report. The implications of this research are considerable, not only for our own project but for the many efforts directed toward information retrieval and other language-related tasks which require bases of semantically-related words.

The ring-structure version of the VIA program has the capacity for the discovery of mutliple relationships among words. In fact, this capacity is so great that to show all the relationships the program discovers results in almost unusable quantity of output for the human investigator. The figure already cited in this report -- 600 pages to show the relationships among words in chapter one of Soviet Military Strategy which are associated with the word aggression -- makes that fact abundantly clear. It may well be that we'll want a computer program to sort out and reduce to manageable proportions the information gathered by the ring-structure VIA program. It is necessary first, though, to decide what kind of relationships are useful for given purposes and what kind of summaries would retain necessary information and still be comprehensible. Over the next few months, we mean to experiment with various forms of output in an effort to determine whether further computer procedures are desirable or whether an output structure can be devised which will meaningfully display the information which can be made available by various forms of the ring-structure VIA program.

Because our programs may often be used interdependently, various procedures in VIA must be "adaptively maintained" as other programs are added to the language analysis package. For example, the addition of some statistical procedures may require information (such as number of word types, sentence length frequencies, etc.) best obtained when a given text is being indexed. Obviously, it is more efficient to add counters to the index program than to make a separate run on the text for a parameter such as sentence length. In fact, several different counts are currently made during the indexing program, others are made during the root-grouping procedure, etc. Programming so that such tasks are performed concomitantly not only reduces the number of complete passes required for any given text (Soviet Military Strategy has about 180,000 words) but also makes available more information after any given program run within VIA; such information can sometimes be helpful when deciding what procedure should next be employed. Given these considerations, the first four programs in VIA -- INDEX, SORT, PREFIX and SUFFIX are beginning to emerge as general text preparation programs which can serve as points of departure for the construction of text-specifi thesauri, for studies of statistical properties, or for one or another mapping procedures. Work during the coming year will certainly entail continuing maintenance and modification of these programs as more analytical power is given to the entire language-analysis package of programs.

A major thrust of such program expansion will be in the direction of the non-verbal representation, or description, of texts provided by statistics and by forms of MAPTEXT. As has been suggested earlier in this report, we feel that such representations will gain considerably in power when used in combination with complementary procedures -- correlation analysis and MAPTEXT used in tandem can provide information that neither can offer separately. It is our plan to add many more parameters to our store of statistical information so that a wide choice of possible representations will be available. The major task will be to determine optimum combinations of procedures in general, if possible, as well as for given texts.

For several of our currently available programs, an interactive mode would be a useful addition. MAPTEXT is a prime candidate for an interactive capacity because its purpose is to provide visual displays of information. Obviously, it would be useful to experiment, interactively, with the displays, asking that some elements in a given display be deleted, that others be added, that the representation symbols be modified, etc.

William Buttelmann's ring-structure VIA has also been designed so that it may, when the support is available, be used interactively; the researcher would specify search keys as well as mode and type of search and see displays of the results. In response to what he learned, the researcher could alter the mode or type, or suggest new search keys. As a first step toward providing interactive capacity, Arthur Coston has written a number of PL/I macro processors for a CC-30 display terminal. As time and personnel permit, we will build further on these efforts.

II. PROGRAM DOCUMENTATION

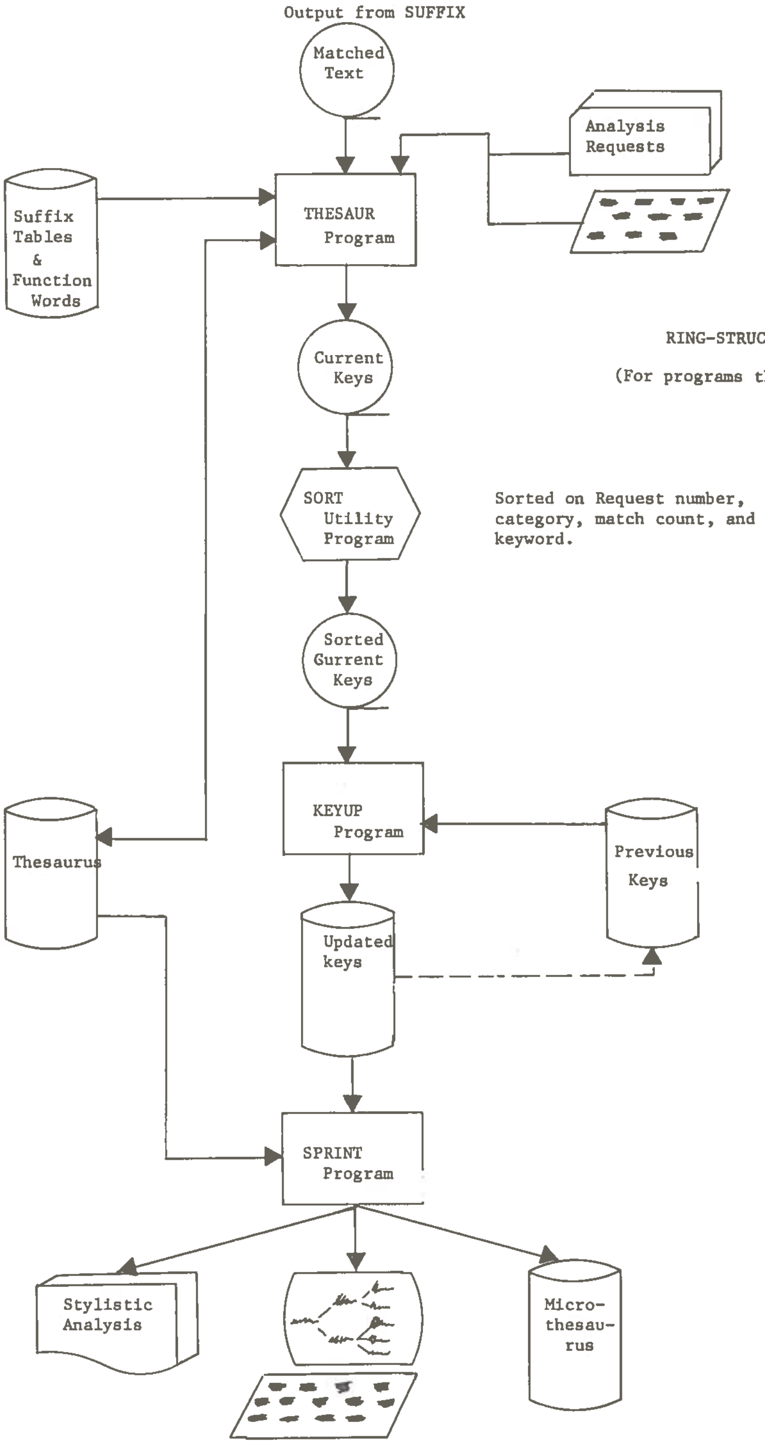
The program documentation actually consists of two sections: the verbal descriptions and instructions for users in this section and program flow charts and listings in the Appendices. Some of the listings, notably those for the text-specific thesaurus program in the list-structure version of VIA and for the first PL/I MAPTEXT program, contain their own documentation. Section II, therefore, contains no separate descriptions for those programs.

Ring Structure VIA - User's Manual

by
H. William Buttlemann

A. INTRODUCTION

For an overview of this system, the reader may refer to section I.B.4., "Ring-structure Version of VIA" of this annual report. Figure 3 gives the overall flowchart of the system. Sections B and C of this manual describe the data with which the user must be most familiar: the control cards supplied by the user, which govern program operation, and the printed output. Section D provides a detailed description of the data sets of VIA, and Section E gives a detailed description of the programs. Appendix B gives a complete computer listing of all VIA programs, with self-documenting comments in the code, the Job Control Cards for running the system, and the printout produced by each program for a sample text and thesaurus. Appendix C gives a number of utility programs for use with VIA.



RING-STRUCTURE VIA

(For programs through SUFFIX, see p. 124)

Figure 3

B。 CONTROL CARDS

Three types of control cards are required for a complete VIA processing run: a TEXTSECT card, one or more ANALYSIS request cards, and a GO AHEAD/RESTART card。 All three are read by the program THESAUR。

All control cards have a "free-format" syntax。 That is, there are no column restrictions on the entries in the cards。 Except for the identifying entries (TEXTSECT, ANALYSIS, GO AHEAD, and RESTART) which must be the first entries in their respective cards, parameters may appear in any order。 The only overall syntax requirements are that entries must be separated by commas or blanks, and the last entry in each TEXTSECT and ANALYSIS card must be followed by a semicolon。

Since the parameters on these cards control all the functions of VIA, an understanding of their meaning is essential to the successful use of the capabilities of the system。

1。 TEXTSECT Card:

This card identifies the current section of text being analyzed。 It is mandatory and must be the first card。 The only entry required on it is:

TEXTSECT = n 。

n is a decimal integer which is the number of the current section of text。 The parameter:

MSGPARM = 'LIST'

may be entered to request the printing of warning messages generated by the THESAUR program。 The default for this parameter is 'NOLIST', which suppresses message printing。

A count of the message that would be printed is given at the end of the THESAUR program.

Optionally, one may enter the string of characters in the remainder of the card following the semicolon. These characters will be printed as a heading at the top of the first page of output.

Examples: TEXTSECT = 1, MSGPARM='LIST';

TEXTSECT = 50; SECTION 50 of JOYCE: PORTRAIT.5/9/69

2. <u>ANALYSIS Request Cards</u>:

Each card supplies the parameters for a complete text analysis and thesaurus search and print. The number of ANALYSIS cards that may be entered for a run is limited by the size of the REQUEST table in the THESAUR program, which is currently 100 cards. The first entry on each card must be:

ANALYSIS n .

n is an arbitrary number identifying the analysis and will be used to identify all printout associated with this analysis request. The other parameters are TYPE, MODE, THRESHOLD, CAT, WORD, DEPTH, and KEYLIST. Only TYPE is mandatory.

TYPE = 'n' .

n is the number 1, 2, 3, or 4. This parameter specifies the type of procedure used to identify key words in the text.

TYPE='1' - <u>Frequent categories</u>. Every category occurring in the text more than a specified number of times is to be used to key a search. The number of times to be used for the threshold, must be specified by a THRESHOLD parameter in the same card. The program will total the number of occurrences in the text of every word in each thesaurus category. If a word occurs in more than one category, its total will be added to each. Every category whose total number of occurrences is equal to or greater than the threshold, will be used to key a search. This type of analysis is lengthy, but enables the system to choose significant content in the text, even though it is not identified by the high frequency of any particular word, because the significance is based on the high occurrence of categories.

TYPE='2' - Frequent roots. Every root occurring in the text more than a specified number of times is to be used to key a search. The number of times to be used as the threshold must be specified in the THRESHOLD parameter. The program will total the number of occurrences of every MATCNT in the text. If the total for a MATCNT is equal to or greater than the threshold, every word in the MATCNT will be used to key a search. This is done by generating a KEY entry for each word in the MATCNT. This type of analysis is somewhat faster than type 1. The system chooses significant content in the text based on the frequency of word roots.

TYPE='3' - Category. The category must be specified by a CAT parameter in the same card. It will be used to key a search. No count considerations are used. This type of analysis is much faster than the previous types and is useful for searching for relationships to a particular category, however obscure.

TYPE='4' - Word. A particular word must be specified by a WORD parameter. It will be used to key a search. This type of analysis is as fast as TYPE 3, and is useful for searching for relationships to a particular word, however obscure. The word need not be in the text, but if not, must be in the thesaurus. Thus, for example, this type of analysis, combined with search mode D, may be used to find all words in the text related to the parameter card.

The MODE parameter:

MODE = 'x'

specifies the mode of thesaurus search used in the SPRINT program. x must be one of the letters A, B, C, D, or E. If the MODE parameter is omitted or incorrectly specified, mode A will be taken by default and a message printed to that effect. The modes are best described in terms of the tree-shaped output showing the word and category relationships.

The five modes are:

MODE = 'A' - Text limited. All nodes are in the text. As the program searches down a path of related words in the thesaurus, it will abandon that path as soon as it encounters a word not in the text. The path down to that point will be printed.

MODE = 'B' - Text oriented. Root and leaves are in the text. If the root is not in the text, nothing is printed. If it is, each path is pursued until no new words can be found that are in the text. The path is printed up through the last textual word encountered. Thus, this word becomes a leaf of the tree. Intermediate nodes may or may not be in the text.

MODE = 'C' - Text rooted. Root in the text. Only the root is required to be in the text. Each path is printed down through the number of levels specified by the DEPTH parameter. This kind of search is used to look for words, in or out of the text, that are related to the root, a word or category in the text.

MODE = 'D' - Text related. Leaves in the text. This mode is similar to mode B, except that the root is not required to be in the text. This kind of search is used to look for words in the text related to a given word or category, whether or not it is in the text.

MODE = 'E' - Subthesarus. The whole subthesaurus rooted at the key is printed, down through the depth specified in the DEPTH parameter. Nothing is required to be in the text.

The THRESHOLD parameter:

THRESHOLD = 'n'

is only used in type 1 and type 2 analyses. n specifies the frequency threshold.

The CAT parameter:

CAT = 'category'

is only used in type 3 analyses. The category specified becomes a search key.

The WORD parameter:

WORD = 'word'

is only used in type 4 analyses. The word specified becomes a search key whether or not it is in the text. The word must be in the thesaurus, or no relationships will be found and no output will appear.

The DEPTH parameter:

DEPTH = 'n'

specifies the depth of search that is to be made in the thesaurus. n must be an integer 1 through 9. Search depths greater than 9 will be reduced to 9 by the syntax checking routines of THESAUR, and a message to that effect will be printed. If the DEPTH parameter is omitted or incorrectly specified, a depth of 3 will be taken by default and a message to that effect will be printed.

The KEYLIST parameter:

KEYLIST = 'option'

is used to specify whether a listing of the keywords for this analysis is requested. The options are LIST and NOLIST. NOLIST is default. If LIST is specified, THESAUR will print a listing of all keys generated for the ANALYSIS request, before they are passed to the program KEYUP.

Examples of ANALYSIS request cards:

```
     ANALYSIS 1, TYPE = '2',MODE = 'D', THRESHOLD = '100',
                          DEPTH = '4';
     ANALYSIS 2, TYPE = '1', THRESHOLD = '50';
     ANALYSIS 3, TYPE = '4', MODE = 'D', DEPTH = '9',
                         KEYLIST = 'LIST', WORD='MIND';
     ANALYSIS 4, TYPE = '3', CAT = '100.1';
```

3. GO AHEAD/RESTART Card:

This card concludes a batch of ANALYSIS Requests. It must have either the words "GO AHEAD" or "RESTART", beginning in column 1 of the card. Nothing else may be entered on it. "GO AHEAD" signals VIA to begin processing the ANALYSIS Requests that have been submitted. "RESTART" causes THESAUR to cancel all ANALYSIS requests that have been submitted and to attempt to read a new batch of control cards. A GO AHEAD card is mandatory if it is desired to continue with the rest of VIA processing. If no GO AHEAD card is present, the program will respond as if a RESTART card had been entered.

C. PRINTED OUTPUT

The purpose of VIA is to find and display verbally indexed relationships. These relationships are printed by the program SPRINT in a tree-structured format. The root of the tree is the key which generated the search. Each level of the tree corresponds to a new depth of search in the thesaurus, and is printed in a different column on the computer listing. The root is level zero. Since words are linked to other words by membership in a common category, and categories are linked together by containing the same word, the printout has alternate columns of words and categories. The further to the right one reads on the page, the more remote one is from the keyword. The further apart two words are in depth, the more remote is their association.

All words which do not appear in the text are printed enclosed in parentheses. When a word appears for the first time in a section of text, it is proceded by a dashed line (--------). Keys (roots) which are being printed because they have appeared as a key in some earlier section of text, but do not qualify as keys in the current section, are preceded by an asterisk. A key appearing for the first time as a key is preceded by a period. To aid and encourage batching analyses, all SPRINT output is printed in order by ANALYSIS Request number.

There are five modes of pruning the printed tree. These are described in detail under the MODE parameter in the section "ANALYSIS Request Cards." Appendix B gives a sample printing for each mode and for each type.

Normally the printed tree is single-rooted, the one root being the search key. However, there is one situation in which the tree will have more than one root. If the search key is a word which is a "temporary" entry in the VOCAB (a word in the text which was not in the original thesaurus, but which has been added to the thesaurus because other words with the same root, i.e., same MATCNT, were found in the thesaurus), then the "temporary" word is associated with the rest of the thesaurus through the "permanent" words with the same root (MATCNT). It is actually those "permanent" words which key the search-and-print algorithm. Thus, when the SPRINT program encounters a search key which is a "temporary" word, it first finds all the permanent words in the VOCAB with the same MATCNT and then initiates a thesaurus search-and-print for each. Each of these words is printed in the column labelled "SEARCH KEY", beneath the temporary word which is the original search key, and from each branches a tree. Thus, the whole tree is multiple-rooted, there being as many roots as there are permanent words with the MATCNT of the original search key.

The only other printouts from VIA are self-descriptive editing messages, debugging aids, and keylists produced by the THESAUR program.

Figure 4 is a sample printout from SPRINT. The key is the word, MEMORY, and is in two categories, 501.1 and 100.1. The only other word in category 501.1 is MIND, which is also

```
ANALYSIS     2 - MODE B,  TYPE 4.  SEARCH KEY IS "MEMORY            "
                 SEARCH DEPTH = 3, BY DEFAULT
----------------------------------------------------------------
| SEARCH KEY      LEVEL 1          LEVEL 2          LEVEL 3     |
|WORD CATEGORY  WORD  CATEGORY  WORD  CATEGORY  WORD  CATEGORY  |
----------------------------------------------------------------

. MEMORY
        501.1
        ---------- MIND
                        501.2
                        --------- PROCESSES
                        --------- PROCESS
                                      100.2
                                      706.0
        100.1
                   (COMPUTER       )
                          100.2
                          ------- PROCESSES
                                  PROCESS
                                      706.0
                                      501.2
                                      ------- MIND
                                (PROGRAM         )
                                      706.0
                                      ------- PROCESSES
                                      ------- PROCESS
                   (PROGRAM        )
                          706.0
                          ------- PROCESSES
                          ------- PROCESS
                                      501.2
                                      ------- MIND
                                      100.2
                          100.2
                          ------- PROCESSES
                          ------- PROCESS
                                      706.0
                                      501.2
                                      ------- MIND
```

Figure 4

in 501.2 along with PROCESSES and PROCESS. Category 100.1 contains two other words, not in the text: COMPUTER and PROGRAM. PROCESSES, PROCESS, and PROGRAM are linked to MEMORY through COMPUTER. PROCESSES and PROCESS are also linked to MEMORY through PROGRAM, by means of two categories, 706.0 and 100.2, which represent two different senses of PROGRAM. MEMORY and MIND have one direct association, in category 501.1, and three remote associations via three different paths through the thesaurus.

Text:

1. Input text. The input text is a simple character stream, in the format of standard printing conventions.

2. Formatted text. The INDEX program produces a reformatted text, with index information. Each record has the following format:

WORD LENGTH	MATCNT	FREQUENCY COUNT	WORD
4	4	4	1-58

3. Matched text. The SUFFIX program produces a text data data set consisting of 1 record per type from the original text. Each record has the following format.*

WORD LENGTH	MATCNT	FREQUENCY COUNT	WORD
2	5	5	18

*In all record format diagrams, the numbers below each field specify the length of the field in storage positions (bytes) as represented in the IBM S/360.

VOCAB

A vocabulary, which is an alphabetical list of each word type in the thesaurus, together with text information and a pointer to the directory.

DRCTRY

A directory, which is a list of categories occurring in the thesaurus, the initial position of each category, and its length (in number of entries) in the thesaurus. The directory is ordered on category.

THES

A thesaurus, which is a list of pairs of pointers, one representing a word in the vocabulary and the other representing a thesaurus category; the thesaurus is ordered on category pointer.

To manage these data sets, VIA does its own input-output buffering. Logically, each data set is a vector, each element being one record. The data sets are stored on disk in blocks of records. Records and blocks are referred to by their index number within the data set. Zero-origin indexing is used throughout. Thus, the computation of the location of record $\underline{n}$ is straightforward: Let $\underline{b}$ be the blocksize of the data set. Then record $\underline{n}$ is in position $\underline{p}$ of block $\underline{m}$, where $m = \text{floor}(n/b)$ and $p = b|n$.

4. VOCAB. The vocabulary is a direct-access data set which contains a record for each word in the thesaurus and for each word in the text not in the thesaurus. The latter are inserted by the UPDATE_VOCAB section of THESAUR during its processing of a particular text section. THESAUR always deletes these record types left by previous runs on a different text. Thus, they are not permanent members of the master thesaurus. They will, however, become permanent members of any microthesaurus for a specific text. In addition, each record contains information necessary for thesaurus searching and enough text information to eliminate the need for further text searches. The text information is inserted by THESAUR. Each record has the following format:

VMATCNT	VSECT	VDIR@	VCOUNT	VX	VFLAG	VWORD
4	4	4	4	1	1	18

VMATCNT - The matchcount developed and inserted by SUFFIX. It serves a dual purpose here:

1. To identify the root.
2. To note that this word or a temporary entry with the same MATCNT is in the text.

VSECT - Number of the first section of text in which this word appeared.

VDIR@ - Directory pointer. Index in the DRCTRY of the first category containing this word.

VCOUNT - Frequency count. Total number of tokens of this word in the current section of text.

VX - Empty.

VFLAG - Used to indicate that there are other words in the VOCAB with this same MATCNT that are also in the text. The flag is 1 if this and others with the same MATCNT are in the text; 2 if others, but not this, are in the text; 3 if this is a new or temporary addition to the VOCAB from the text; and 5 if this is the last record in a VOCAB bucket and the entries overflow into the overflow bucket.

VWORD - Eighteen characters are presently allowed for the word.

The VOCAB has a bucket-type organization, for faster searching.* There is a bucket for every pair of English letters that appears in the Thesaurus, and an additional overflow bucket -- 677 possible buckets. The key transformation is by means of table lookup based on the leading pair of letters in the word. Bucket size and block size are independent, so there may be more than one block per bucket. Location information is kept in two 26x26 matrices, VKEYS and VEXTS, stored in the THSCTL data set. For example, let "xyz" be a word and let x be the ith letter in the alphabet and y the jth letter. Then, each entry VKEYS(i,j) gives the number of the first block in the VOCAB bucket containing the word "xyz", and each entry VEXTS (i,j) gives the number

*For a description of table organization, searching, and key transformations, see Brooks, F. P., Jr., & Iverson, K. E., Automatic Data Processing, John Wiley & Sons, (New York), 1963. Section 7.3.

of additional blocks in the bucket. Thus, if the program needs to find "MEMORY" in the VOCAB, it looks up VKEYS(13,5) and VEXTS(13,5). Suppose VKEYS(13,5) = 42 and VEXTS(13,5) = 2. Then the program will start a search for "MEMORY" in block number 42 of VOCAB, and will scan through block number 44. If VKEYS(13,4) = 0, then there are no entries in VOCAB beginning with "ME."

5. DRCTRY. The directory is a direct-access data set which provides the linking between the Vocabulary and the Thesaurus. There is one record for each thesaurus category. Each record has the following format:

DCAT	DTHS@	DLNG
8	4	4

DCAT - Category Number. The classification number of this category. Any combination of symbols, 8 characters or less is acceptable - such as the designations in Roget's Thesaurus (e.g. 101a.3).

DTHS@ - THES Pointer. Position of the first record of this category in the Thesaurus.

DLNG - Length (in # of entries) of this category in THES.

6. THES. The actual thesaurus is a direct-access data set. Each record represents an entry in the thesaurus -- that is, an entry of one word in one category. The format is:

TDIR@	TVOC@
4	4

TDIR@ - DRCTRY pointer. Index in the Directory of the next
category containing this word.

TVOC@ - VOCAB pointer. Index in the Vocabulary of the word
for this entry.

For a given word, the Directory pointers thus provide the linking which "chains" through all the Thesaurus entries for that word. The pointer for the final entry of each word is set to point to the Directory entry for the first category containing the word. Thus, each chain is cyclic: a "ring structure." The entire thesaurus, then is a structure of interconnected rings.* Searching is done by stepping around each ring, and for each entry in a ring, stepping around the ring linked with the first one by that entry. Because the rings are closed chains, each search eventually returns to its starting point in the ring. By saving the address of the starting point, one knows when the search is complete.

*For a discussion of chained data representation, see Brooks, F. P., Jr., and Iverson, K.E., Automatic Data Processing, John Wiley & Sons, New York (1963) Section 6.2.

The following simplified diagram may serve to illustrate the organization and linking of the Vocabulary, Directory, and Thesaurus. Suppose we have the following Thesaurus:

Category	100.1	Computer, memory, program
	100.2	Computer, process, program
	501.1	Mind, memory
	501.2	Mind, process
	706.0	Process, program

Then the data sets would be (we have reordered the Vocabulary records and elided unnecessary information for clarity):

VOCAB

	WORD	VDIR@
0	COMPUTER	0
1	MEMORY	0
2	MIND	2
3	PROCESS	1
4	PROGRAM	0

DRCTRY

	CAT#	DTHS@
0	100.1	0
1	100.2	3
2	501.1	6
3	501.2	8
4	706.0	10
5	99999	12

THES

	TDIR@	TVOC@
0	1	0
1	2	1
2	1	4
3	0	0
4	3	3
5	4	4
6	0	1
7	3	2
8	2	2
9	4	3
10	1	3
11	0	4

As an example, consider a search for the word PROCESS. We first look it up in VOCAB. Its Directory pointer indicates that it first occurs in the second category in DRCTRY. We go to the second entry in DRCTRY and find there category 100.2. Its Thesaurus pointer indicates that this category begins at position 3 in THES. Indeed, the entry for PROCESS is the entry "3-3" at position 4 in the Thesaurus, which is the second entry in category 100.2. To find the other entries for PROCESS, we proceed as follows: the Directory pointer at the first PROCESS entry is 3, indicating that the next entry for PROCESS is in the fourth category. We go to the Directory and find that this is category 501.2, which begins at record 8 in the Thesaurus and is 2 entries long. Indeed, there is an entry for PROCESS in record 9 in the Thesaurus. The Directory Pointer there points to category number 4, which, according to the Directory is number 706.0, begins at entry 10 in the Thesaurus, and is 2 entries long. Entry 10 is the one for PROCESS, and its Directory pointer indicates category 1, the second category, which was the first category in our ring. Thus, we have returned to our starting point, and the search around the ring is complete.

One may ask why the TDIR@'s do not point directly to the next entry in the Thesaurus, rather than back to the Directory. The answer is that the category numbers are needed for printing and, more important, for each category, we suspend the step to the next link in the ring and initiate a ring search on every word in the category. Thus, we need to scan each category in THES, starting at its first entry.

7. THSCTL. There is one additional data set associated with the Thesaurus. This set contains location information for the buckets in the VOCAB, and the blocksize and extent information needed by the VIA programs to do the data set: VBLKSIZE, DBLKSIZE, TBLKZISE, VLASTBLK, DLASTBLK, TLASTBLK, VKEYS, and VEXTS. The values for the first six variables are recorded in data-directed format and give the block sizes in number of records and the

number of the last block of each of the data sets, VOCAB, DRCTRY, and THES. VKEYS and VEXTS are 26 by 26 matrices stored in edit-directed format, one blank and three digits per element.

8. <u>KEYS</u>. This file is a list of words, each of which initiates a search of the Thesaurus. In addition to certain keying and searching information, each key contains the entire VOCAB entry for the key word. This information eliminates a later search of the Vocabulary. Each record has the following format:

KSECT	KDIR@	KVCOUNT	KVFLAG	KFLAG	KMODE	KTYPE	KCOUNT	KUDC@	SORT KEY			
									DRQ#	KCAT	KMATCNT	KWORD
4	4	4	1	1	1	1	4	4	4	8	4	18

KSECT - VSECT from VOCAB.

KDIR@ - VDIR@ from VOCAB.

KVCOUNT - VCOUNT from VOCAB.

KVFLAG - VFLAG from VOCAB.

KFLAG - An asterisk here denotes that this key has appeared in a previous section of text, but does not qualify as a key in the current section. A period indicates that this is the first section of text in which this word qualifies as a key. Otherwise, this field is blank.

KMODE - Search mode from the ANALYSIS request card.

KTYPE - Analysis type from the ANALYSIS request card.

KCOUNT - Frequency threshold from the ANALYSIS request card.

KVOC@ - Location in the VOCAB of this word.

KDEPTH - Search depth limit for SPRINT, from the ANALYSIS request card.

KRQ# - ANALYSIS request number from the ANALYSIS request card.

KCAT - Category designator from which this word was taken. If this information is not pertinent (as in certain types of search), this field is left blank.

KMATCNT - MATCNT from VMATCNT.

KWORD - The key word.

E. Programs.

The main line programs of VIA are, in order of processing, INDEX, SUFFIX, PREFIX, THESAUR, KEYUP, and SPRINT (See Figure 3). In addition, there are a number of utility programs. INDEX, PREFIX, and SUFFIX are documented elsewhere in this or earlier reports. The utility programs are self-documenting; their program listings are given in Appendix C. This section, together with Appendix B, provides complete documentation for THESAUR, KEYUP and SPRINT.

THESAUR

The data sets associated with this program are:

1. TEXT, the formatted text output from SUFFIX, with MATCNTs and frequencies entered. (input).
2. The Thesaurus data sets VOCAB, DRCTRY, THES, and the control data set, THSCTL. (input, update).

3. The control cards submitted by the user; TEXTSECT, ANALYSIS Requests, and GO AHEAD/RESTART card. (input).
4. KEYS, the keywords passed by KEYUP to SPRINT, to initiate Thesaurus searches. (output).

THESAUR edits ANALYSIS requests and puts them in a table for later reference, updates the Thesaurus with the current section of text by inserting information from the text in the VOCAB data set, and creates the data set, KEYS. There are four major sections in the program: REQUEST, INITIALIZE_VOCAB, UPDATE_VOCAB, and BUILD-KEYS. The subroutines of THESAUR are RWRVOC, READDIR, READTHS, TRIPLBREAK, which are internal, and STEM, the major external subroutine of the program SUFFIX.

REQUEST reads the deck of control cards (TEXTSECT, ANALYSIS, and GO AHEAD/RESTART), edits them, and builds the table, REQUESTS, from the ANALYSIS cards. During this process, IRQMAX is used to index the REQUESTS table. Thereafter, IRQMAX retains the position of the last request entered. If a GO AHEAD card is read, processing passes to SORTRQS, which sorts the REQUESTS table on TYPE. Then, a listing of the requests is printed. Next, a search of the table is made for any TYPE 1 or 2 requests.

This completes the REQUEST section.

Before passing the current text against the VOCAB, it is necessary to do a certain amount of cleanup, because it is possible that information from other sections of text or from a previous run of the current section may still be in VOCAB.

This information must be removed. All MATCNTs and COUNTs must be erased, since these may have changed from previous runs. All other information that has been entered from text sections subsequent to the current section is erased. Information from previous sections of text (except MATCNT and COUNT) is retained. This completes the work of the INITIALIZE_VOCAB section.

The next section, UPDATE_VOCAB, passes the text against the VOCAB and enters textual information in the VOCAB. Hereafter, references to textual information may be made directly to the VOCAB, and no further passes of the text are required.

First, however, the code labelled KTS produces a special KEYS record which has the current TEXTSECT number in it and a sort key that will cause it to be sorted first and retained by KEYUP. This is a method of passing the TEXTSECT number to SPRINT and avoids requiring that information to be duplicated by the user.

The text is read at the statement labelled GETT. If a text word is already in the VOCAB, its MATCNT and COUNT are entered. If the VSECT field = -1 (indicating that the word is appearing for the first time) the current TEXTSECT number is entered. If a text word is encountered which is not in the VOCAB, the word is temporarily saved in the table, NEWTWORDS. Whenever there is a change in the first three letters of the text words, there is a pause in the processing, and the subroutine TRIPLBRK is called. TRIPLBRK attempts to link the

NEWTWORDS with words in VOCAB that have the same MATCNT. Since the VOCAB words may not have their MATCNTs entered, it may be necessary to call STEM to attempt to match words. To minimize the number of STEM calls, the NEWTWORDS are first sorted on MATCNT. Then a search of VOCAB for each MATCNT is conducted. If a certain MATCNT is not found already in VOCAB, then the program begins comparing the first NEWTWORD in the MATCNT group with each word in the VOCAB having the same leading triplet, using STEM to check for root sameness. As soon as a match is found, the MATCNT is entered in the VOCAB entry. Once an entry in the VOCAB is found with the same MATCNT, all the NEWTWORDS with that MATCNT are tagged for subsequent entry by setting their NFLAGs to '4'. If no matching entry in the VOCAB can be found, a sequence of messages is printed identifying the NEWTWORDS and stating that no relationships for them can be established in the thesaurus. After the entire table of NEWTWORDS has been processed, those words marked with NFLAG = '4' will be inserted in the VOCAB bucket as "new" (or "temporary") entries. Such entries are inserted in the extra space provided at the end of the bucket. Overflows are placed in the special overflow bucket at the end of the VOCAB. Each of these entries is marked "temporary" by setting its VFLAG to '3'. Finally, the flag in each permanent VOCAB entry with a matching MATCNT is set to indicate that there are temporary entries for this MATCNT: the VFLAG is set to '1' if the permanent word is also in the text, and to '2' if it is not.

Whenever there is a change in the first two letters of the text words, additional work is required if there are TYPE 2 requests. TYPE 2 requests call for summing the COUNTS of all words with the same MATCNT. This work is done in the code labelled PAIRBRK, at the end of the TRIPLBRK subroutine. If there are any TYPE 2 requests, then as the text is passed, each new MATCNT encountered is entered in the table T2TBL, and its total COUNTS is summed. PAIRBRK processes this table and then reinitializes it for the next bucket. T2TBL is compared with all TYPE 2 requests, and a key word record is written in the KEYS data set for each word in each MATCNT whose total satisfies a request threshold. This is done for each TYPE 2 request.

When the entire text has been passed, the work of UPDATE_VOCAB is finished.

The final section, BUILD_KEYS, generates KEYS records for TYPE 1, 3, and 4 requests. If there are TYPE 1 requests, the counts of each category are summed. The sum is compared with the threshold of each TYPE 1 request, and a KEYS record is generated for each category, for each request whose threshold is satisfied by the category sum. For each TYPE 3 request, a KEYS record is generated for the specified category. For each TYPE 4 request, a KEYS record is generated for the specified word.

KEYUP

The data sets associated with this program are:

1. NEWKEYS - the KEYS produced by THESAUR, and sorted. (input).
2. OLDKEYS - the output data set from the previous run of KEYUP on the previous section of text. (input).
3. CURKEYS - the merged and slightly edited keys to go to SPRINT. (output).

KEYUP merges the new key words produced from the current section of text with those from all previous sections: Records only in the OLDKEYS are flagged with an asterisk in KFLAG to indicate that the word is not a current key. Records only in the NEWKEYS are marked with a period in KFLAG to indicate that the word is appearing for the first time as a keyword. KFLAG is printed by SPRINT immediately preceding the keyword.

SPRINT

The data sets associated with this program are:

1. KEYS - merged keywords from SPRINT. (input).
2. The Thesaurus data sets VOCAB, DRCTRY, THES, and the control data set, THSCTL. (input).
3. Analysis results printout. (output).

SPRINT searches the Thesaurus for associations and prints out the word-category relationship pattern in a tree format. SPRINT reads the KEYS file sequentially; each key causes one complete search-and-print operation, the key word or category serving as the root of the tree. (For a description of the output and the search modes, see Section C, "Printed Output.")

The program consists of a main procedure and a recursive subprocedure, WORD. There are also subprocedures, TEMPRNT, to seek and print temporary entries in VOCAB, and PGHDG, to print page headings. The recursive subprocedure, WORD, conducts one complete scan around a ring of categories (constituting a word). Within WORD is a loop (the coding labelled CAT) which conducts one complete scan around the ring of words (constituting a category) for each category in the ring. Within the CAT loop is the recursive call, at the coding labelled NXTLVL. SPRINT keeps track of the level of recursion in LEVEL, and WORD will continue to call itself until LEVEL reaches the value of the DEPTH parameter.

Three data areas of interest are the vectors, PATH, WORDSP, and CATSP. PATH is a vector of pointers representing the current path the search-and-print algorithm is working down. Because of the interlocking ring structure of the Thesaurus (word ring-category ring-word ring-category ring, etc.) the odd-numbered nodes will be word pointers and the even-numbered nodes will be category pointers. The purpose of PATH is to provide a means for enforcing the overall printing rule of eliding any node (and the subtree rooted at it) that has already appeared in the path.

WORDSP and CATSP are vectors whose elements are character strings. The strings are the actual words and category numbers to be printed. These vectors are needed, because, for some search-and-print modes, the decision to print cannot be made when a node is found. When a decision to print is finally made, these vectors provide the necessary print information, and rereadings of the VOCAB and DRCTRY are eliminated. These vectors are organized so as to make their entries correspond with the entries in PATH. Thus, WORDSP(I) = VWORD(PATH(I)) and CATSP(I) = DCAT(PATH(I)). As a consequence, half of each -SP vector is unused, a small expense of memory to save the time that would otherwise be needed to compute their indices: only one index serves for all three vectors.

Other data items of interest are: CURCAT, the index in DRCTRY of the first category. This location marks the starting point of a word ring search; PRINTNDX, which indicates the last position in PATH (and hence in WORDSP and CATSP) that has been printed.

B. PREFIX

by
John B. Smith

BACKGROUND

Project VIA's need of a computational procedure for determining the presence of an English prefix on a word has both immediate and far-reaching implications.

In order to determine patterns of inter-relations among content carrying words, it early became apparent that procedures would have to be developed that could "recognize" or group together words with the same root or stem. Part of this task was accomplished by SUFFIX, which groups together words of the same root form but with different suffixes. PREFIX accomplishes the other half of the task. It allows us to note the presence of a concept or idea carried in the root of the word but modified and masked by the prefix. Thus it has immediate use in the VIA package.

Although syntactic analysis is of no immediate concern for VIA, recent computational studies have indicated the importance of affixes as indicators of part-of-speech. This consideration led to Resnikoff's and Dolby's work on operational definitions of affixes and an algorithmic approach to determining affixes.* Their work has been followed up by Lois Earl in her attempts to assign part-of-speech categories by rules based primarily on affixes and internal vowel clusters.** Unfortunately, her goal of 95% accuracy has been attained only hypothetically because of errors in her dictionary, and her work is restricted to a corpus of only some 20,000 words. PREFIX, on the other hand, is defined over a considerably larger corpus, the unabridged _Random House Dictionary_.

*H.L. Resnikoff and J.L. Dolby, "The Nature of Affixing in Written English," _Mechanical Translation_, VIII (1965), 84-89. Also "The Nature of Affixing in Written English Part II," _Mechanical Translation_, IX (1966), 23-33.

*Lois L. Earl, "Automatic Determination of Parts of Speech of English Words," _Mechanical Translation_, X (1967), 53-67.

Consequently, PREFIX may have important implications in syntactic studies that lie outside the immediate concerns of Project VIA.

GENERAL APPROACH: Essentially, PREFIX's approach is a table look-up procedure, but without the disadvantage of costly time consumption of multiple searches through the entire table. An extensive list of admissible English prefixes was complied by consulting available lists of affixes and by consulting our working dictionary. We placed two linguistic restrictions on prefixes:

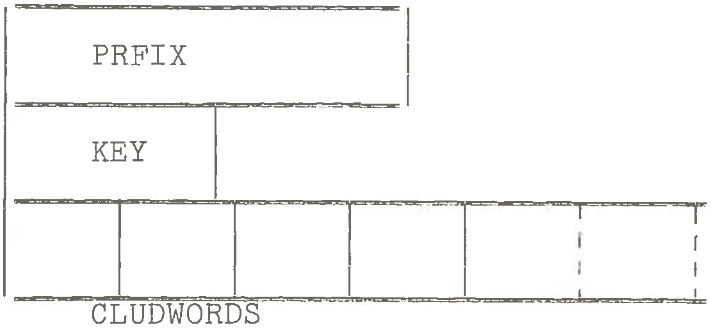
1. The prefix must be a bound morpheme.
2. A word is considered to have a prefix only if the remainder of the word, without the prefix, is independent, i.e., not a bound morpheme.

After preparing our list of prefixes, we next had to account for words whose initial letters are identical with given prefixes but which are not prefix-carrying words. For example, at, although beginning with a, is not a prefix-carrying word; atypical would be. SUFFIX functions by having lists of exceptions. However, we found such an approach impractical for many prefixes. The a prefix is an example of this problem: an exception list would involve most of the words beginning with a listed in our dictionary. One solution to the problem is to use an inclusion list and consider only those words on the list as having legitimate prefixes. Such an approach would work well for the prefix a, but not for in. Our ultimate solution was to compile either an exception list or an inclusion list for each prefix, dependent upon which list would have fewer members. A note on problems of specific work selection will be given later in this paper.

PREFIX: As pointed out above, PREFIX is a table look up procedure; however, since text input is assumed to be in logical records, one word per record, and the records to be in alphabetical order, the look up time can be reduced to a minimum. In fact, the task can be accomplished with just one complete pass through the prefix lists. Each prefix is loaded into a PL/I structure along with its accompanying list of words and the key that specifies whether the list is an inclusion list or an exclusion list. This structure has the following format:

```
91  PTABLE (35),
    02  PRFIX CHARACTER (8),
    02  KEY FIXED DECIMAL (1),
    02  CLUDWD (300) CHARACTER (18);
```

or



for one prefix with accompanying CLUD list. The structure, PTABLE, will accommodate 35 prefixes, each with as many as 300 accompanying words.

Since there are obviously more than 35 prefixes in the English language, we had to resort to an overlay approach to "roll in" and "roll out" the appropriate prefix lists. This task is performed by a call to a subroutine called PFETCH.

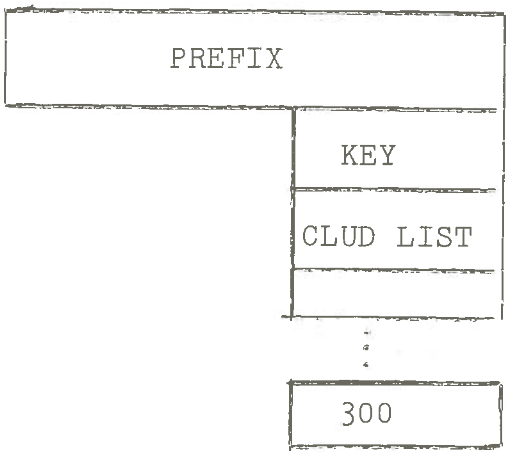
PFETCH: This subroutine reads a sequential data set of prefixes with accompanying lists--hence referred to as CLUD lists or CLUD words--and loads them into the structure PTABLE. This is done for all prefixes beginning with the same letter of the alphabet. When a prefix is read in that begins with a different letter, it is stored temporarily, and execution falls into some "housekeeping" tasks which will be explained later. Control then passes back to the main procedure. For example, the first call to PFETCH will load in all _a_ prefixes, with CLUD lists, until the first prefix beginning with a _b_ is read. Prefixes, like text-word records, are in alphabetical order as are their CLUD lists.

MAIN PROCEDURE: The main procedure is controlled by a large DO-loop for which each value of the indexing variable represents a letter of the alphabet. Incoming text records are first tested against the control letter of the alphabet. Processing continues so long as the first letter of a text word matches the control letter; if not, PFETCH is called to load in the next group of prefixes. [The text word is next checked to see if it is identical with the preceding word that was just processed. If so, it is either processed or rejected as was the preceding word. If the word is different then it falls into a series of tests.]

First the word is tested to determine its length. If the word has fewer than four characters, it is rejected (REJECT is set equal to the word so that the next word read in can be tested against it). This is done on the assumption that words with three and fewer characters do not contain admissible prefixes. We have not found exceptions to this rule in any tests yet processed.

If the word is longer than three characters it is tested against the list of prefixes. If the prefix is of length N, the first N letters of the word are checked for a match. If these conform, then a check of the accompanying CLUD list is performed. The word is checked against the words of the CLUD list until a match is found or the word is no longer further along in alphabetical sequence than the remaining words in the CLUD list.

PTABLE: for each prefix.



If the word is found to match one of the words in the CLUD list, then the prefix key is consulted. If the key is 0--indicating an exclusion list--the word is rejected, REJECT is set equal to the word, and a new word is read in for testing. If the key is 1--indicating an inclusion list--a duplicate record, except for the omission of the prefix, is created. LSTWORD is set equal to the word indicating that a valid prefix was found for subsequent testing, and a new text word is read in. When a prefix match is found, the location of the prefix within the PTABLE structure is noted, and similarly for a match within the CLUD list. Since the text words, prefixes, and CLUD lists are all in alphabetical order, subsequent tests for text words can begin with the prefix and CLUD word last found to match a text word. The prefixes and CLUD lists are processed in their entirety only once, thus greatly reducing look up time. The time gained, however, by passing through the list of prefixes only once is not without some qualifications.

This last point can best be developed by an illustration. The word *atypical* contains a legitimate *a* prefix, but in alphabetical sequence it would come after words with *ab* prefixes, *ad* prefixes, etc. If we wish not to keep searching the prefix lists, prefixes that admit such words must be flagged. It turns out that each such prefix is "contained in" the prefix immediately following it. That is, the "troublesome" prefix will be shorter in length than the succeeding prefix and will match it letter-for-letter for its length. Some such prefixes are _a_ (contained in _ab_), _arch_ (contained in _arche_), etc. The task of flagging each such prefix is performed in PFETCH. The locations in PTABLE of all prefixes of this kind are loaded into an array called PERMFIX, with space for ten prefixes (actually what is stored is a number pointing to the location of the prefix in PTABLE--thus for _a_ the pointer would be '1').

It will be recalled that we tested each text word for a match with a prefix of N letters. When a match was found, the prefix was marked and subsequent testing began there. If the prefix does not match the first N letters of a word, a test is made to see if the word follows the prefix in alphabetical sequence. For example, if testing for the word _aftermath_ begins with the prefix _ad_, a mismatch of the first two letters with the prefix will occur. _Aftermath_ will then be seen to come after _ad_ in alphabetical sequence; consequently control will shift to the next prefix, and so on until a match is found or the word precedes the prefix in sequence. At this point testing will shift to the group of prefixes that admit words in later sequence than words with the next lower prefix--as was the case with _atypical_. The word is tested against all such prefixes--referenced through the pointers in PERMFIX-- and their associated CLUD lists. If a match of both prefix and CLUD word is found, a duplicate record is formed or not depending upon the key. If a match of prefix but not CLUD word is found, a duplicate record is formed if the key is 0 (indicating that the list is an exclusion list). Either

REJECT or LSTWORD is set equal to the word accordingly.

PRINT: PRINT is a subprocedure that does the actual processing of the prefix. In the present experimental version of PREFIX the prefix is lost; however, in the functioning version it will remain as a separate entry within the logical record for each text word. PRINT is called whenever an additional record is to be created. Into the sequential data set is introduced a duplicate record but with the word stripped of prefix. A listing on the printer is also made for manual reference. Format is identical to input format and is as follows:

1	3	9	12	19
2	6	3	7	18
↑ LENGTH OF RECORD	LIN.#	PG.	BLANK	WORD

The actual removal of only the prefix is accomplished by using the VARYING character attribute of PL/I. By storing the prefix in a location with this attribute, the computer records the actual length of the record contained (in this case the prefix). Consequently, the portion of the text word without the prefix can be picked off by using the SUBSTRING operator. The second operand, the position of the variable (in this case the text word) at which the substring is to begin, is set equal to the length of the particular prefix plus 1. After each call to PRINT, processing continues as before.

TABLE PREPARATION: Scientific and obsolete words, proper nouns, and multiple word idioms are not included in the CLUD lists; however, words marked "archaic" that might appear in literary texts are included. The problem of accounting for forms of words to be included but with variant suffix forms was solved in the following way. Once we determined that a root form was to be included in the list, we made the entry conform to only those letters that the variant forms share in common. Thus the CLUD list entry for complete, completely, completing, etc. would consist of the letters complet. This approach is applicable only when the entry form excludes all words not of the same root and which are not to be included in the CLUD list. This constraint necessitated our marking certain short words as complete in themselves. For example, add is included in the form 'ADD '; otherwise, the program would assume that the add entry would include all words with these first three letters.

At present, the program is operational, but we are in the process of making corrections and additions to our CLUD lists to account for unforeseen omissions and inclusions. One of our working hypotheses is that the prefix is much more fundamentally involved with the semantic content of a word than the suffix; but it also appears much less frequently than the suffix within English texts. However, our experience with PREFIX is limited and initial assumptions may well be modified later.

C. CONTEXT

by

John B. Smith

1. CONTEXT is a package of programs that attempt to show the way in which the important terms of a natural language text inter-relate and combine to form the larger substantive themes of that text. The primary emphasis of VIA in the past has been to define major themes by finding and laying out lists of related terms that appear in a document or segment of a document using various Thesauri.* To use an analogy, this process might be considered similar to discovering and displaying the various "bones" that are present in the structure or "skeleton" of substantive ideas that underlie a text. CONTEXT attempts to show how these various segments or "themes" fit together, how they are inter-related. The patterns of inter-related terms not only mark strong stylistic characteristics but also are quite revealing as to the actual content of the piece.

More specifically, CONTEXT looks at small subsections of text (the size of the subsection is determined by the user) to see whether or not any of a list of the most "important" words of the text are present. Such a list or lists is provided by VIA as output, and may be used as CONTEXT input

*(For a detailed description of VIA see S.Y. Sedelow, _et al._, _Automated Language Analysis_, 1967-1968).

if the researcher chooses. The program then examines all such subsections to determine the consistence with which various words are used together. The factors or patterns that emerge are quite indicative of the way in which the author puts together individual words or ideas to form larger themes. These factors are determined by using a standard Principal Component Analysis program.* (Similar "canned" factor analysis procedures are available at most computation centers). Input into this program is a list of numbers, or matrix, where the numbers in each row represent the number of occurrences of each term being examined in a particular subsection of text. There are as many rows of numbers as they are subsections of text.

*The remainder of Section II.C is intended for the reader with very limited mathematics. Therefore, the presentation is intuitive and highly analogous. For a more detailed and rigorous mathematical treatment see Harry H. Harman's Modern Factor Analysis, University of Chicago Press, 1967, 2nd Revised Edition.

Factor analytic studies of word clusters have been successfully conducted in several other fields of research. Many of the social and behavioral science journals carry articles concerning such studies; some that might be of particular interest to the reader are The American Journal of Psychology, Educational and Psychological Measurement, and Psychometrika. One effort that warrants specific reference is that of Drs. Howard Iker and Norman Harway. While at the University of Rochester Medical School they used techniques quite similar to those of CONTEXT to examine the patterns of associated ideas in transcribed psychotherapy sessions. (See "A Computer Approach Towards the Analysis of Content," Behavioral Sciences X, # 2 (4/65), pp. 173-182).

$$\begin{array}{l|ccccc}
 & \text{word}_1 & \text{word}_2 & \text{word}_3 & \ldots\ldots\ldots & \text{word}_m \\
\hline
\text{section}_1 & f_{11} & f_{12} & f_{13} & \ldots\ldots\ldots & f_{1m} \\
\text{section}_2 & f_{21} & f_{22} & f_{23} & \ldots\ldots\ldots & f_{2m} \\
\text{section}_3 & f_{31} & f_{32} & f_{33} & \ldots\ldots\ldots & f_{3m} \\
\vdots & \vdots & \vdots & \vdots & & \\
\text{section}_n & f_{n1} & f_{n2} & f_{n3} & \ldots\ldots\ldots & f_{nm}
\end{array}$$

Thus if one is interested in M different words or "subthemes" and divides the text into N subsections, then the matrix is MxN and there are N·M individual elements in it. The factor analysis program* looks at each pair of words in all subsections and assigns the pair a value ranging from -1 to +1 (this value is called a correlation coefficient). If the terms consistently occur together in the same context the correlation coefficient would be near +1; if the terms never occur in the same environment the correlation coefficient would be near -1; a random occurrence of the terms with regard to one another would result in a correlation coefficient near 0. Thus the NxM matrix reduces to a square MxM matrix called the correlation matrix.

*CONTEXT, as I have stated, uses what is actually a principal component procedure; however, I shall use the more general term, factor analysis, in referring to this data reduction technique.

$$\begin{array}{c|ccccc}
 & \text{word}_1 & \text{word}_2 & \text{word}_3 & \circ\;\circ\;\circ\;\circ\;\circ & \text{word}_m \\
\hline
\text{word}_1 & a_{11} & a_{12} & a_{13} & \circ\;\circ\;\circ\;\circ\;\circ & a_{1m} \\
\text{word}_2 & a_{21} & a_{22} & a_{23} & \circ\;\circ\;\circ\;\circ\;\circ & a_{2m} \\
\text{word}_3 & a_{31} & a_{32} & a_{33} & \circ\;\circ\;\circ\;\circ\;\circ & a_{3m} \\
\circ & \circ & \circ & \circ & & \\
\circ & \circ & \circ & \circ & & \\
\circ & \circ & \circ & \circ & & \\
\text{word}_m & a_{m1} & a_{m2} & a_{m3} & \circ\;\circ\;\circ\;\circ\;\circ & a_{mm}
\end{array}$$

Here it is probably easiest to understand the process if we switch to a geometric or vector model. One may regard each row of the correlation matrix as a set of numbers ordered by their position (a_{11}, a_{12}, ----, a_{1m}, etc.), or as a point in a Euclidean space of dimension M, or as a vector. If one regards each row as a vector, then the set of all M vectors (one for each row) will generate a space of dimension D, such that $D \leq M$.

For example, the three vectors

$$\left(\begin{array}{ccc} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{array}\right) \begin{array}{l} \text{alpha}_1 \\ \text{alpha}_2 \\ \text{alpha}_3 \end{array}$$

could be said to generate the usually three dimensional Euclidean space

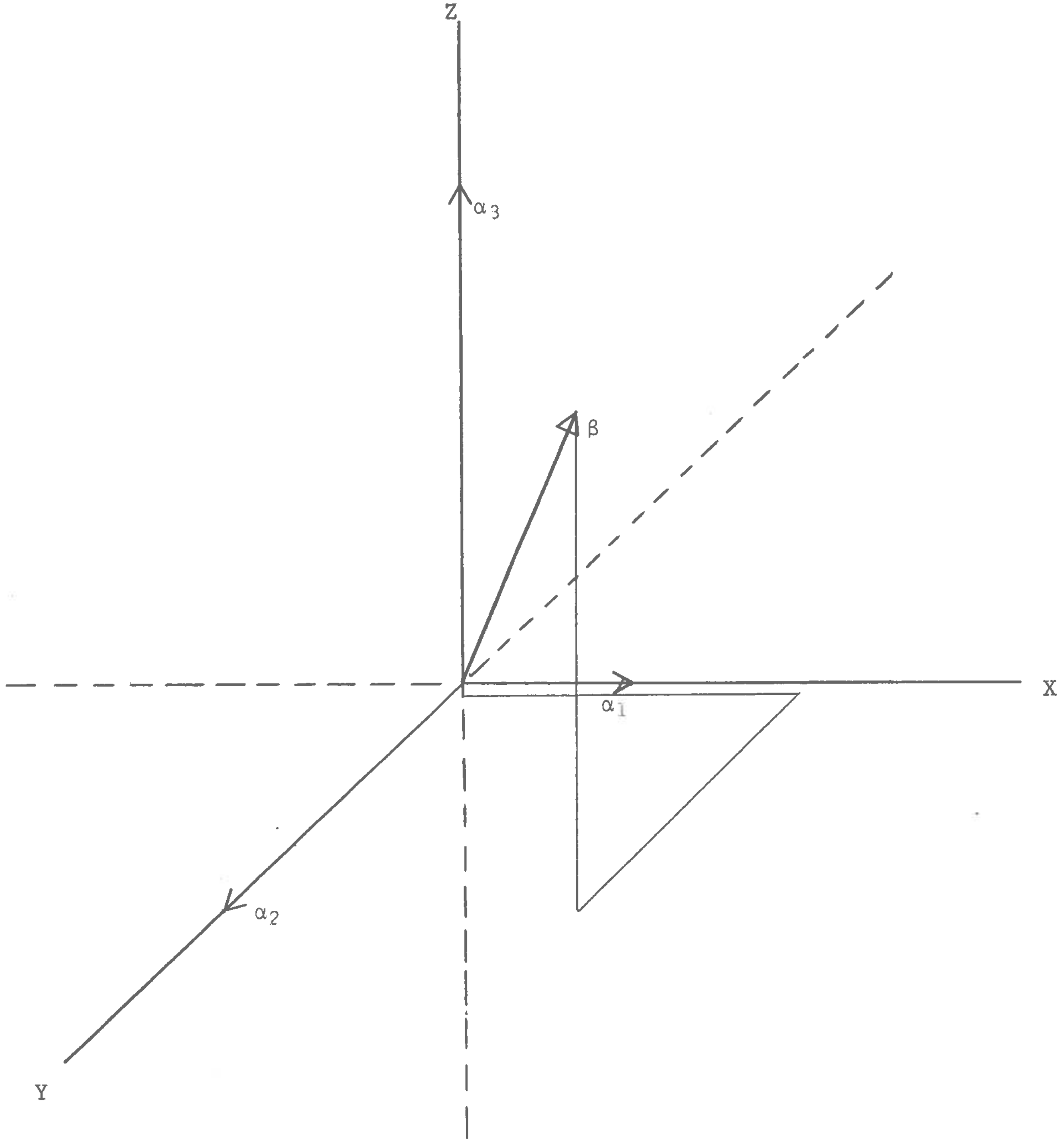


Figure 5

since any point or any vector in the space could be generated by taking linear combinations of the three vectors given. For example $\beta = (2,2,3)$ can be represented by $2\alpha_1 + \alpha_2 + \alpha_3 = 2(1,0,0) + (0,2,0) + (0,0,3) = (2,2,3)$. In general, then, N vectors will generate a space of dimensionality less than or equal to N.

The factor analysis model seeks a group of vectors, formed by various combinations of the original vectors, that comes closest to generating the original space of the correlation matrix. This approximation is close when a number of original vectors lie relatively near to one another. In 2-space this process might be represented as follows:

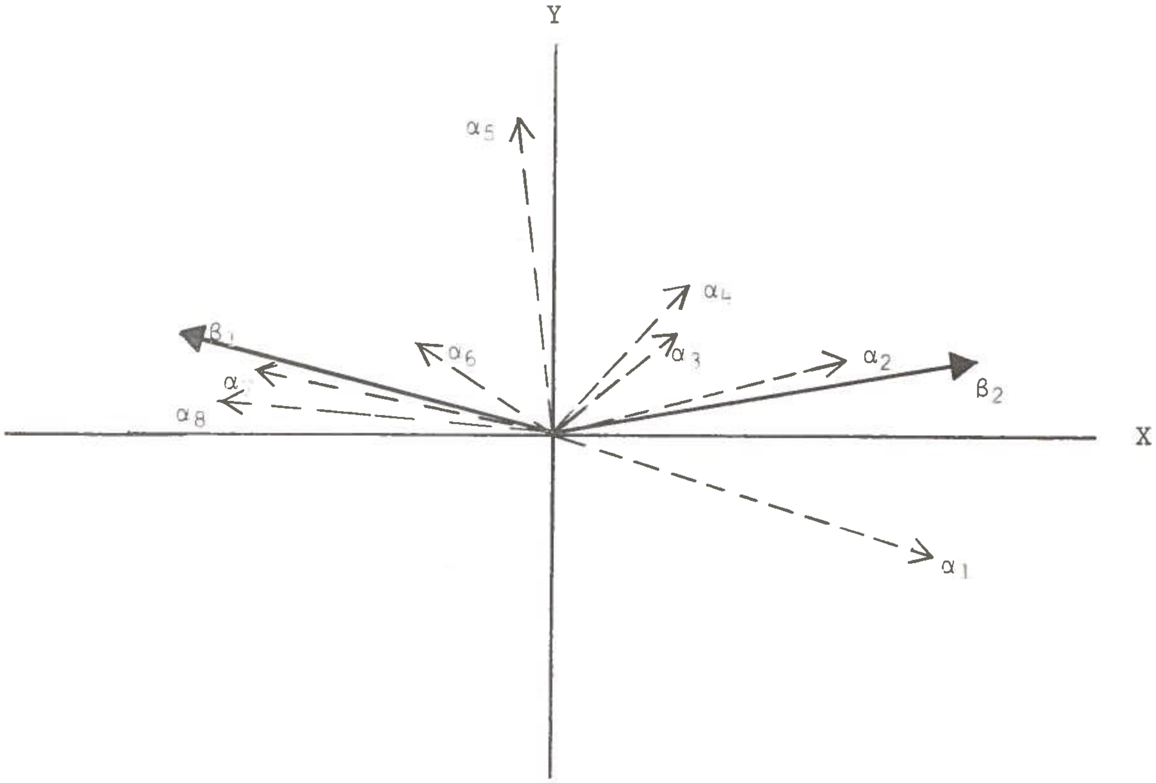


Figure 6

where the original vectors, $(\alpha_1, \alpha_2, ---, \alpha_8)$ might be approximated or reduced to β_1, β_2, with α_5 probably left over.

What the program actually produces is a set of column vectors (or factor loadings) of the form

word_1	a_{11}
word_2	a_{21}
word_3	a_{31}
.	.
word_n	a_{n1}

Each element or weight of the factor represents the degree to which the particular variable (word in our case) contributes to the vector. Thus individual factors can be thought to be most strongly characterized by those variables or words which contribute the largest weights. A negative weight implies the absence of that variable (or word) in conjunction with the variables or words that most strongly characterize the factor.

For example:

word_1	.05
word_2	.91
word_3	.63
word_4	.81
word_5	.53
word_6	.66
word_7	.32
word_8	.21
word_9	-.55
word_{10}	.11

This factor is most clearly defined in conjunction with words 2, 4, 6, 3, 9, and 5; however, word 9 consistently _does not_ appear in context with the others.

2. Each factor in the CONTEXT model represents an interrelation of words or smaller themes that consistently occur together. Thus each might be thought of as a characteristic "large" theme of the document. We are currently testing CONTEXT on several texts: James Joyce's _A Portrait of the Artist as a Young Man_ and the Praeger translation of _Soviet Military Strategy_. In the first chapter of the _Portrait_, for example, almost from the very beginning verbal motifs are introduced that play beneath the texture of the entire novel. One such motif concerns fear and retribution associated most immediately with birds and Stephan's eyes. The motif is introduced by the refrain

> Pull out his eyes,
> Apologise,
> - - - - - - - - - -

This same note of tension and fear prevades the chapter. Stephan is sent to a Jesuit school at Clongowes where he spends a period in the infirmary, goes home for Christmas, and returns to school. Upon his return a significant series of events begins. Stephan accidently breaks his glasses; because he cannot see to write he is unjustly punished with a pandybat. At the end of the chapter Stephan escapes the tyranny of Authority. What is most significant about Joyce's narrative is not just the series of events but the patterns of associations that they raise for Stephan. These patterns of association are, of course, both a strong stylistic feature of Joyce's writing style as well as a substantive aspect of the content of the novel. Many of these patterns of associations are reflected quite strongly in the factors developed by CONTEXT.

The motif of fear mentioned above can be seen in the following factor:

#4

Apologise	.826
eyes	.807
out	.671
player	.268
light	.260

⋮

And the stay in the infirmary can be seen in the following:

#5

brother	.919
Michael	.891
queer	.541
infirmary	.294
call	.290

in which Stephan is most impressed by the unusual ("queer" is his word) habits of the attending Jesuit, Brother Michael.

Several factors reflect the events leading up to the pandybat episode:

#8		#17	
broke	.790	Father	.779
glass(es)	.783	Arnall	.766
write	.379	write	.402
did	.294	want	.286
home	.256		

These factors can be seen to reflect the association between the broken glasses and writing (it was in Father Arnall's class where Stephen was "pandied"). The punishment itself can be seen most graphically in factor #3:

pain	.800
pandybat	.712
sound	.712
loud	.569
down	.352
hand	.343
feel	.331
differ	.277
felt	.230
quick	.211

The very interesting aspect of this factor is that the pain of the pandybat striking Stephan's open hand is explicitly identified with the loud sound that the bat makes. This identification among various senses is both a strong stylistic feature of Joyce's writing as well as an important substantive element of The Portrait. A number of other factors can be seen to reflect various aspects of the novel, but these illustrations should give the reader some idea of how CONTEXT currently functions.

3. Specific Programs of CONTEXT:

For convenience, CONTEXT has been referred to in this paper as a single program. Actually it consists of several individual procedures or programs, some of which are slight modifications of programs already in the VIA package. The normal VIA indexing program, INDEX, had to be modified slightly to facilitate the picking out of the immediate environment of words. INDEX, originally set up to allow convenient manual cross-referencing, establishes sequence in terms of numbers for volume, chapter, paragraph, sentence, and word, for prose; CONTEXT uses a modified form of INDEX, called LINDEX, in which words are merely numbered linearly--the first word is numbered 1, the second 2, etc. until the end of the text. This indexing scheme *greatly* simplifies the task of determing the "distance" between words. After indexing, the text is sorted alphabetically using the standard S/360 sort package. The sorted records are then fed into a suffixing program that groups words according to root. Thus, *complete*, *completely*, *completing*, etc. would be grouped together. All such forms are identified by a unique, five digit number called a MATCHCOUNT; however, the word itself is left as it originally appears. The words are again sorted, this time on matchcount and secondarily on the indexing sequence. The text then goes through an updating program, STATEX, in which the frequencies attached to each record are updated to correspond to the total frequency of all tokens for the same MATCHCOUNT. Also, STATEX computes the mean, standard deviation, and prints out a distribution table. The final data preparation step occurs in PRETEXT in which all words occurring over a certain frequency (expressed either in absolute terms or in terms of m + n(s.d.)) are selected for context study. This list is edited against both inclusion and exclusion lists that are furnished by the user. PRETEXT then computes a large matrix in which each row lists the frequency of occurrence of each of the words within a particular text segment (100 word, 500 words, or whatever unit the user specifies). Finally, this matrix is fed into a standard principal component program for analysis.

LINDEX

LINDEX is an indexing program based upon INDEX, described in detail in the annual report for 1967-68. It differs primarily in that the indexing information attached to each record is a six digit, sequential number. By indexing on linear sequence (as opposed to volume, chapter, paragraph, sentence, word, etc.), one can easily determine numerical units (50 words, 100 words, etc.) for designating subsections of text or word environments. This capability is essential for examining environmental correlation and for content analysis. (See section on CONTEXT).

INPUT: Input must be in 80 character records (usually punched cards or magnetic tape). The only restriction placed upon the text is that it be blank delimited. That is, every unit, including punctuation, that the user wishes the computer to recognize must be separated by blanks from other units. (Example: . . . must be separated by blanksΔ.). Thus the program will take prose, poetry, speech transcription, etc., so long as it is blank delimited. Shift characters (such as ">", "]", or "<") may be used to indicate different type fonts, stage directions, etc.; however, if the user wishes to have these deleted from the data, he must list them individually in statement 4 of LINDEX.

OUTPUT. Output consists of fixed length records, one word or punctuation mark per record.

FORMAT.

1 3	9	12	19	36
LN.	LN.#	PG.	FILL	WORD
2	6	3	7	18

The data set may be either put on tape directly or passed to a temporary storage location (usually a 2314 disk pack), sorted alphabetically, and then put onto tape. (Again, with slight modification of Job Control Language, data may be passed directly to subsequent programs such as PREFIX and SUFFIX.)

MAIN PROGRAM. A cardimage is read into a 1 x 80 array called CARDIMAGE. A subprocedure, FORM, then begins at column 71(cols. 72-80 are reserved for page numbers, sequence numbers, etc.) and concatenates letters until a blank is found.

No provisions are made for word continuation from one card to the next. Therefore, if a word cannot be completed by column 71, blanks should be left at the end of the card and the word should be punched on the subsequent card. This word is returned to the main procedure. Accompanying each record is the page number of the word (for manual reference) along with a six digit linear sequence number. The latter is incremented by one for each new word returned.

ALPHABETICAL SORT: The sort used is one of the standard sorts of the System 360 sort package that is called through Job Control Language. For a detailed description of the options available, one should consult the IBM sort-merge manual. Records are sorted on the field beginning in column 19 of each record, for a field of 18 characters, with the sort in ascending order.

SUFFEX

SUFFEX is a slightly modified form of SUFFIX, described in detail in Automated Language Analysis: 1967-1968.

This form differs in the following respects. The original program, SUFFIX, printed for each word-token the indexing information, but the data set passed to later programs dropped this information. Thus the passed data set became a type (not token) data set with one entry for each unique word-type in the text. The modified version, SUFFEX, differs in that it takes an "exit" at the print statement of the older program and creates a record for each token that is passed either on tape or disk to later programs. The record format is as follows:

1 3	9	12	17	21	38
LENHTH	LINEAR SEQ. #	PAGE	MATCH COUNT	FREQ	WORD
2	6	3	5	4	18

Thus the output is essentially an "updated" data set with matchcounts and frequency of word-type added to the records.

STATEX

STATEX is an interface program that runs between the suffix program and CONTEXT. The frequency counts attached to each record in SUFFEX were frequencies of word type; the updating of these counts so that they represent the total frequency of all word tokens for a root or matchcount (not just a word type) is done in STATEX.

In addition to updating these counts, STATEX computes the mean and standard deviation of frequencies of a text data set. For data sets with similar distributional patterns, thresholds may be set in terms of mean + n(s.d.). By doing this, the user is unable to set thresholds proportionally for data sets of different sizes. STATEX also keeps track of the number of roots (actually matchcounts) for each frequency interval. This information is printed out in table form which may in turn be used in regression procedures that "fit" a curve to the data.

Our preliminary results imply that the relative locations of various vocabulary items within the patterns for various texts would provide parameters for determining stylistic variations as well as content. Similarly the distributional patterns themselves might be useful in such analyses. For example, the distributional pattern for a fairly small section of text taken from Joyce's A Portrait of the Artist is a rather close approximation of a negative binomial expansion. If the distributional patterns for various texts can be approximated by standard statistical functions, then the defining parameters might well serve as author and content discriminators. However, let me emphasize that our research in this direction is just beginning and any results that we have at this point are tentative.

INPUT:

It is assumed that the text data set has been processed by SUFFEX and that records are in matchcount order. (See discussion of SUFFEX above).

OUTPUT:

Output is identical to input except that frequency counts have been updated so that the frequency represents that of all tokens with the same matchcount. For example, if there are 28 occurrences of complete, 16 of completely, and 4 of completed, each record of these variant forms of the same matchcount would be updated so that the frequency carried would be 48. This process would facilitate the selection process for programs making searches based on frequency thresholds. There is additional printed output of data described in the discussion of the main programs.

MAIN PROGRAM:

Records are read into a structure until they differ in matchcount. The structure,TEMP, has room for 500 records. Each element is of the form:

```
(01 TEMP (500))
          02  JUNK    CHARACTER (11)----holds portion of record,
                                        not used in STATEX, that
                                        must be passed to PRETEXT.
          02  MATCH   FIXED DECIMAL (5)-Matchcount number
          02  FREQ    FIXED DECIMAL(4)--frequency of each word
                                        type
          02  WORD    CHARACTER(18)----text word
```

When a matchcount does not correspond with the previous matchcount, processing falls into the main execution loop. Beginning with the first entry in TEMP a check is made of subsequent pairs of words. When a mismatch occurs, the frequencies of the two words are added together since the frequencies attached to each record are the frequencies of that word type, not matchcount.The process continues as long as the matchcounts are the same. When the process is completed, the total frequency is placed in the FREQ slot for all tokens and the records are put out onto tape or disk. At this time several other counters are incremented. The formula used for computing the standard deviation is a function of the total frequency of all tokens and also the *square* of the number of tokens of each matchcount. Therefore, several running totals are kept: one, of total number of tokens, is incremented by merely adding the frequency count to the previous total; the sum of frequency squared is si milarly incremented, but by adding to the previous sum the *square* of the frequency for the matchcount. To facilitate computing the mean, a count of the number of unique matchcounts is also kept.

The frequency distribution that is computed is of the following form:

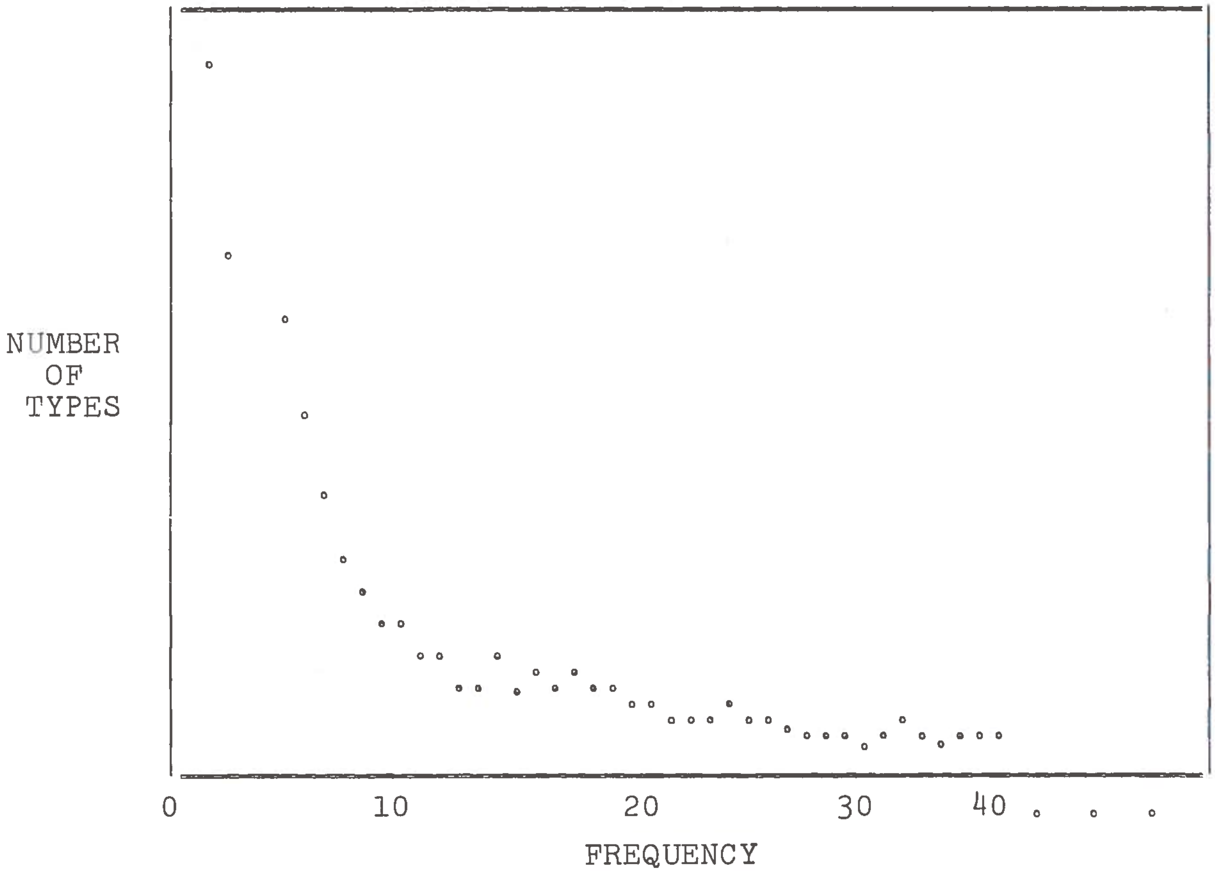


Figure 7

Each point represents the *number* of matchcounts for each unit frequency; i.e. the number of matchcounts that occur once, twice, etc. This information is kept in a 1 x 2000 array.*

On endfile, the program computes the mean (mean = freq. ÷ number of types) and standard deviation.

$$S.D. = \frac{\sqrt{N(\sum(freq.)^2 - (freq.)^2}}{N}$$

Thresholds for CONTEXT searches can then be expressed in terms of these statistics and passed to PRETEXT. Finally, the program prints the table of distributional data.

PRETEXT

PRETEXT is the main data preparation program in the CONTEXT sequence. It is the program that computes the data matrix that is actually passed to the "canned" principal component program.

INPUT: Data Records are the updated output of STATEX, with frequencies denoting total tokens per matchcount. It is assumed that records are sorted on matchcount, word, and linear number, all in ascending order.

Also passed from STATEX are a threshold frequency computed from the formula Threshold (CUT = mean + *n*(standard deviations) for a user specified *n*, the maximum number of tokens for a

*I assume that *no* matchcount will have a frequency count greater than 2000. Thus for each computational cycle, the counter in the array (called T) corresponding to the total frequency of the matchcount is incremented by one. (In the example of *Complete* etc. above, the corresponding counter would be in T(48).

single matchcount (used in allocating the dimensions of a major storage structure), and the number of matchcounts greater than or equal to the threshold. Also read in are two lists of matchcounts; one an inclusion list, to be used regardless of frequency, and the other an exclusion list, used similarly.

<u>MAIN PROCEDURE</u>:

The inclusion and exclusion edit lists are read into one dimensional arrays. Then records are read in one at a time. If the frequency of the record is greater than the frequency threshold, the program calls the EDOUT subprocedure to make sure that the user has not edited out this word.* If the word is not edited out by EDOUT, it along with the matchcount and its location are loaded into the structure LOCAT.

```
02  WORD CHARACTER (6)
02  MATCH FIXED DECIMAL (5)
02  NLOC FIXED DECIMAL (3)
02  LOC (max) FIXED DECIMAL (6)
    (one slot for each location).
```

A variable, LSTMAT, is set equal to the matchcount so that the next MATCHCOUNT can be tested directly against this variable instead of going through the whole procedure outlined above.

If the frequency of the incoming record is less than the threshold, the EDIN procedure is called. If the MATCHCOUNT is found, the record is loaded into LOCAT and processing continues as above; if not, a variable, REJECT, is set equal to the MATCHCOUNT and all subsequent records with this matchcount

*For example, words like <u>said</u>, <u>here</u>, etc. are of very high frequency in some texts, but of little thematic interest. They are not discarded in the function word edit facility of SUFFEX, but the user may wish to edit them out of the context analysis procedure.

are similarly rejected.

On Endfile a series of processes begins. First the locations under each matchcount are sorted to be sure they are in ascending order. Then a process begins that determines the number of times a particular matchcount appears within a specified environment (for example, within 50 words) of every other MATCHCOUNT. This information, stored in a square matrix called DATAC, is printed out for manual reference. After print out, this storage area is freed.

Finally, the program constructs the data matrix that serves as input into the principal component program. The user specifies the unit of text that he desires (for example, he may wish to divide the text up into 100 word chunks). Each row will have an entry for each word or matchcount selected above. Each entry represents the number of times that a particular matchcount occurs in a particular section of text. The principal component program requires that this data be received row by row. However, it turns out that with a unit of 100 words for, say, 160 matchcounts that the matrix is too large to be held in the computer. Furthermore, it will be seen that it is most convenient to compute the matrix by columns. If this is done, then the matrix cannot be easily manipulated if output is on tape or disk. The problem was solved by a rather interesting technique. Only some 20% of the elements of the matrix are non-zero. Therefore a matrix with the dimension of the matrix to be output is declared, but as a bit matrix.

(Usually a computer byte--that would hold an A, 1, etc.--consists of eight bits--0 or 1). It would have the following form:

```
0   1   0   0   1   .   .   .   .   0
1   0   0   1   0   .   .   .   .   1
1   1   0   1   0   .   .   .   .   0
.
.
.
.
.
.
1   0   0   1   0   .   .   .   .   0
```

Each element may be either 1 or 0.

Next a one dimensional array, LFIELD, is declared Fixed decimal (2) with as many elements as the total number of tokens loaded into LOCAT. It would look like this:

```
 ____
| 01 |
|----|
| 05 |
|----|
| 10 |
|----|
| 12 |
|- - |
|    |
|- - |
|    |
```

Similarly a dummy "column" of the matrix is declared as well as a counter for each column of the matrix.

Finally, processing begins at the top of LOCAT. Each location for a particular matchcount is divided by the unit (say 100). The (result + 1)th location in the dummy column is incremented by one. (For example, word 568/100 = 5 + 1 = 6: the 6th slot of the column is incremented to show that the particular word or matchcount occurs in the 6th unit of text). When all locations for a particular word have been processed, the corresponding elements in the large bit matrix are changed from 1 to 0; and, beginning at the top of the column, each non-zero element is loaded into the long LFIELD array. This process is repeated until all matchcounts are processed. Then the total number of 1's in each column is computed. Thus a 1600 x 1000 matrix capable of holding two digit numbers can be held in approximately 60,000 bytes instead of 320,000.

From this information, the matrix to be processed by the factor analysis program is constructed a row at a time and put out onto tape or disk. Reconstruction follows this form: for element a_{ij}

$$N = \sum_{j=1}^{i-1} \text{COLUMNTOTAL}(j) + \sum_{m=1}^{i} a_{mj} \neq 0 \quad .$$

III. Professional Activities of Project Personnel

Sally Y. Sedelow

Publications:

*Automated Language Analysis, Report on research for the period March 1, 1967 to February 29, 1968, Contract N000 14-67-A-0321, Office of Naval Research. University of North Carolina. DDC # AD 666-587.

*Editorial, Computer Studies in the Humanities and Verbal Behavior, Vol. I, No. 2, August, 1968.

Papers/Seminars/Addresses/etc.:

*Speaker, "Computer-Aided Research in the Humanities,"
University of Notre Dame, March, 1968;
University of Delaware, April, 1968;
Ohio State University, May, 1968;
Pennsylvania State University, November, 1968.

*Paper, "Computer-Aided Research in the Humanities," Advances in Computing, Fourth Stony Brook Conference, June, 1968.

*Paper, "The Computer and the Humanities: A Contradiction?" National Science Foundation Park City Conference on the Computer and Undergraduate Education, September, 1968.

*Speaker, "Computer-Aided Stylistic Analysis," Haverford College, Phillips Fund Lecture, February, 1969.

Activities:

*Member, Advisory Panel, National Science Foundation's Institutional Computing Services Section, 1968 - .

*Chairman, Special Interest Committee on Language Analysis and Studies in the Humanities (SICLASH), Association for Computing Machinery, 1968 - .

*Co-Editor, Computer Studies in the Humanities and Verbal Behavior, 1967 - .

*Reviewer of papers for Fall Joint Computer Conference, 1968, and Spring Joint Computer Conference, 1969.

*Field Reader of Proposals, U. S. Department of Health, Education, and Welfare, 1966 - .

*Chairman, Symposium: Mathematical Approaches to Word-Frequency Phenomena, Psychometric Society Spring Meeting, 1968.

*Invited Participant, Conference on the National Archives and Statistical Research, Washington, D. C., May, 1968.

*Proposal Evaluation, Canada Council, 1968 - .

Walter A. Sedelow, Jr.

Publications:

*"A Quarter-Century Reflected," The Shield, 12 (2), Spring, 1968.

Papers/Seminars/Addresses/etc.:

*"Social Trends and Library Science," Address at the Annual Meeting, UNC School of Library Science Alumni Association, Chapel Hill, April 27, 1968.

*Member, National Science Foundation Site Visit Panel for INTREX, Massachusetts Institute of Technology, May 2-3, 1968.

*Invited participant, Conference on the National Archives and Statistical Research, sponsored by National Archives and Records Service and the National Academy of Sciences, Washington, D. C., May 27-28, 1968.

*Remarks, UNC School of Library Science Alumni Association, American Library Association annual meeting, Kansas City, Missouri, June 26, 1968.

*Session Chairman, Session 9A, "The Computability of Cultural Materials," 1968 Annual Meeting, Association for Computing Machinery, Las Vegas, Nevada, August 29, 1968.

*Panel Chairman, Human Sciences, National Science Foundation Park City Conference on Computers and Undergraduate Education, Park City, Utah, September 8-13, 1968.

*Panel discussion leader, "Special Collections for Colleges and University Libraries," College and University Section meeting, Biennial Meeting of the Southeastern Library Association, Miami Beach, Florida, November 1, 1968.

*"Trends in Library Science Education," Address to the Quarterly Meeting of the North Carolina Chapter of the Special Libraries Association, Chapel Hill, N.C., December 4, 1968.

*"The Computer and Liberal Arts Education," Lecture at Washington College, Chestertown, Maryland, February 28, 1969.

Activities:

*Associate Editor, Social Forces, 1966 - .

*Board of Editors, Computer Studies in the Humanities and Verbal Behavior, 1966 - .

*Series Editor, The Free Press/Macmillan Company, 1968 - .

*Trustee, International Social Science Institute, 1966 - .

*Member, North Carolina Public Library Certification Board, 1967 - .

*Member, University Research Council Sub-committee for the Social Sciences and Professional Schools, University of North Carolina, 1967 - .

*Member, UNC-CH Committee on University Government, 1967 - .

*Consultant, Jacksonville (Illinois) State Hospital, February, 1968 - .

*Referee of Technical Papers, Language Analysis and Studies in the Humanities, for the Technical Program Committee of the ACM annual meeting, 1968.

*Member, Administrative Board, Frank Porter Graham Child Development Research Center, Chapel Hill, North Carolina, February, 1968 - .

*Member, American Council of Learned Societies' Committee on Information Technology, February, 1968 - .

*Staff participant, National Library Workshop for Population Research Center Libraries, Chapel Hill, May 15-16, 1968.

*Participant, planning session for the National Science Foundation Conference on Computers in Undergraduate Education, Park City, Utah, August 1-2, 1968.

*Member, Organizing Committee for the Special Interest Committee on Social Science Computation (SICSOC), 1968 Association for Computing Machinery annual meeting, Las Vegas, Nevada, August 28, 1968.

*Member, Sub-Committee on Professional Schools, Faculty Council Committee on University Self-Study, University of North Carolina, Chapel Hill, 1968 - .

*"Computer-aided Analyses of Interdisciplinary Discourse Barriers," NASA Project # 325-NAS-4-401, Co-principal Investigator (with UNC-CH Space Sciences Committee), 1968 - .

*Member, Steering Committee and Chairman, Section on Informational and Social Aspects of Advanced Technology, UNC Space Sciences NASA Project, 1969 - .

Walter L. Smith

Publications:

*"Necessary conditions for almost sure extinction of a branching process with random environment," Annals of Math. Statist. 6, 2136-2140 (December, 1968).

Papers/Seminars/Addresses/etc.:

*Invited Lecture: "On So-called Complete Convergence of Partial Sums of Random Variables" - delivered at Eastern Regional Meeting of the Institute of Mathematical

Statistics, Blacksburg, Virginia, April 8, 1968.

*Invited Lecture: "Renewal Theory and its Ramifications: A Second Look" - delivered at National Meeting of the Institute of Mathematical Statistics, Madison, Wisconsin, August 27, 1968.

H. William Buttlemann

Publications:

*"Ring-Structure Version of VIA," and "Program Documentation, Ring-Structure VIA," in S. Y. Sedelow, *et al.*, *Automated Language Analysis 1967-1968*, University of North Carolina, 1968, pp. 19-27, 85-105.

Activities:

*National Science Foundation Graduate Traineeship for 1968-69.

*Consultant, Research Triangle Institute-Information Retrieval.

*Consultant, Bruce Payne Associates-Programming Systems.

William G. Hickok

Publications:

*"Documentation for INDEX, SUFUN, and SUFFIX," in S. Y. Sedelow, *et al.*, *Automated Language Analysis 1967-1968*. University of North Carolina, 1968, pp. 35-84, 113-137.

*Master's Thesis: *An Application of the MaGee-Boodman Model for Inventory and Production Control*, 1969, 155 pages.

Papers/Seminars/Addresses, etc.:

*"Programmed Flight Hour Record Report" - documentation and computer program, NAIRU-A2, NARTU NAF Andrews for NARTU Training, February 1969.

Activities:

*Consultant: Administrative Data Processing Department, University of North Carolina, October-December, 1968.

*Consultant: Ohio Furnace Company, Columbus, Ohio, April, 1969.

APPENDIX A

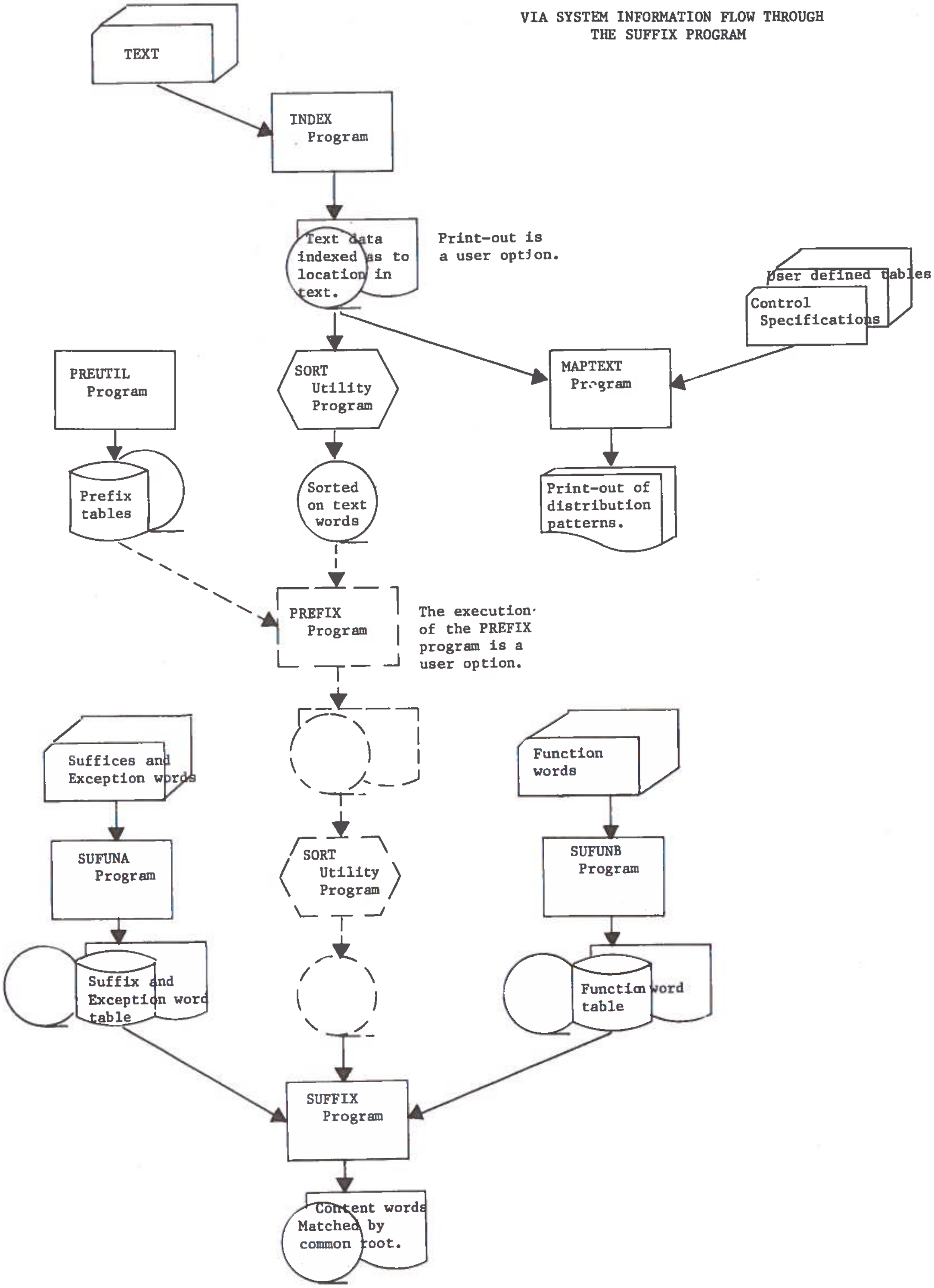
VIA System Information Flow Through the SUFFIX Program

by

William G. Hickok

(For listings of programs other than MAPTEXT and PREFIX), see S. Y. Sedelow, *et al*, *Automated Language Analysis, 1967-68*, pages 112-137. The programs listed in last year's report have been somewhat modified but the overall logic remains the same. Current listings thoroughly documented can be provided for any individual or organization desiring to use the programs.)

VIA SYSTEM INFORMATION FLOW THROUGH
THE SUFFIX PROGRAM


TEXT
INDEX Program
Text data indexed as to location in text.
Print-out is a user option.
User defined tables
Control Specifications
PREUTIL Program
SORT Utility Program
MAPTEXT Program
Prefix tables
Sorted on text words
Print-out of distribution patterns.
PREFIX Program
The execution of the PREFIX program is a user option.
Suffices and Exception words
Function words
SUFUNA Program
SORT Utility Program
SUFUNB Program
Suffix and Exception word table
Function word table
SUFFIX Program
Content words Matched by common root.

APPENDIX B

Ring-Structure VIA Program Listings

(Including Sample Output)

by

H. William Buttelmann

```
  1              THESAUR: PROC OPTIONS(MAIN);
                 /*********************************************************************/
                 /* THIS IS THE MAIN PROGRAM OF VIA.                                  */
                 /* THIS PROCEDURE DOES ALL PREPROCESSING OF REQUESTS, TEXT, &        */
                 /* THESAURUS PRIOR TO THE SEARCH-&-PRINT FUNCTION.                   */
                 /* IT UPDATES THE THESAURUS WITH THE CURRENT SECTION OF TEXT BY      */
                 /* MAKING TEMPORARY INSERTIONS INTO THE VOCAB DATA SET.              */
                 /* IT ALSO CREATES THE DATA SET OF KEYS WHICH KEY THE SEARCH-&-PRINT */
                 /* FUNCTION OF SPRINT.                                               */
                 /* MAJOR SECTIONS:                                                   */
                 /*   REQUEST, INITIALIZE_VOCAB, UPDATE_VOCAB, BUILD_KEYS.            */
                 /* MAJOR SUBROUTINES:                                                */
                 /*   RWRVOC, TRIPLERK, STEM.                                         */
                 /*********************************************************************/
  2                      DCL VOCAB   FILE RECORD DIRECT UPDATE ENVIRONMENT(REGIONAL(1));
  3                      DCL DRCTRY  FILE RECORD DIRECT INPUT  ENVIRONMENT(REGIONAL(1));
  4                      DCL THES    FILE RECORD DIRECT INPUT  ENVIRONMENT(REGIONAL(1));
  5                      DCL THSCTL  FILE STREAM               INPUT;
  6                      DCL TEXT    FILE STREAM               INPUT;
  7                      DCL CURKEYS FILE RECORD SEQUENTIAL OUTPUT;
  8                      DCL   VBLKSIZE        FIXED BIN,
                               DBLKSIZE        FIXED BIN,
                               TBLKSIZE        FIXED BIN,
                               VLASTBLK        FIXED BIN,
                               DLASTBLK        FIXED BIN,
                               TLASTBLK        FIXED BIN,
                               IVOC            FIXED BIN,          /* VOCRCD INDEX     */
                               IDIR            FIXED BIN,          /* DIRRCD INDEX     */
                               ITHS            FIXED BIN,          /* THSRCD INDEX     */
                               IVOCSAVE        FIXED BIN,
                               VKEYR           FIXED DEC(5),       /* PASSED TO RWRVOC */
                                                                   /*   SUBROUTINE.    */
                               DKEYR           FIXED DEC(5),       /* PASSED TO READDIR*/
                               TKEYR           FIXED DEC(5),       /* PASSED TO READTHS*/
                               VKEYSAVE        FIXED DEC(5),
                               VINCOREKEY      FIXED DEC(5)        /* VOC BLOCK IN CORE*/
                                                   INITIAL(-1),
                               DINCOREKEY      FIXED DEC(5)        /* DIR BLOCK IN CORE*/
                                                   INITIAL(-1),
                               TINCOREKEY      FIXED DEC(5)        /* THS BLOCK IN CORE*/
                                                   INITIAL(-1),
                               VKEY (26,26)    FIXED DEC(3),       /* VOCAB BUCKET KEYS*/
                               VEXT (26,26)    FIXED DEC(3);       /* VOCAB BUCKET     */
                                                                   /*   EXTENSIONS     */
  9                      DCL   01 VOCBLOCK CONTROLLED,  /* THESAURUS VOCABULARY BLOCK.*/
                                 02 VOCRCD(0:VBLKSIZE-1),     /* WE DO OUR OWN BLOCKING*/
                                   03 VMATCNT  FIXED BIN,          /* MATCH COUNT:     */
                                                                   /*   ENTERED IFF    */
                                                                   /*   THIS OR ANOTHER*/
                                                                   /*   WITH SAME      */
                                                                   /*   MATCNT OCCURS  */
                                                                   /*   IN TEXT.       */
                                   03 VSECT    FIXED BIN,          /* SECTION OF TEXT  */
```

```
                                                                 /*   WHERE THIS WORD*/
                                                                 /*   FIRST APPEARED.*/
                                  03 VDIR@    FIXED BIN,         /* POINTER TO ENTRY */
                                                                 /*   IN DRCTRY OF   */
                                                                 /*   FIRST CAT WITH */
                                                                 /*   THIS WORD.     */
                                  03 VCOUNT   FIXED BIN,         /* COUNT OF TOKENS  */
                                                                 /*   IN TEXT.       */
                                  03 VX       CHAR(1),           /* ALIGNMENT PADDING*/
                                  03 VFLAG    CHAR(1),           /* ' ' NORMALLY     */
                                                                 /* '1' IF THIS AND  */
                                                                 /*   TEMP ENTRIES   */
                                                                 /*   WITH SAME      */
                                                                 /*   MATCNT ARE IN  */
                                                                 /*   TEXT.          */
                                                                 /* '2' IF TEMP      */
                                                                 /*   ENTRIES BUT NOT*/
                                                                 /*   THIS ARE IN    */
                                                                 /*   THE TEXT.      */
                                                                 /* '3' IF TEMP ENTRY*/
                                                                 /* '5' IF THIS IS   */
                                                                 /*   THE LAST RCD OF*/
                                                                 /*   A BUCKET AND   */
                                                                 /*   THERE ARE MORE */
                                                                 /*   ENTRIES IN THE */
                                                                 /*   OVERFLOW BUCKET*/
                                  03 VWORD    CHAR(18);          /* TYPE ENTRY.      */
10                      DCL   01 DIRBLOCK CONTROLLED,   /* THESAURUS DIRECTORY BLOCK. */
                                02 DIRRCD(0:DBLKSIZE-1),
                                  03 DCAT     CHAR(8),           /* CATEGORY ID      */
                                  03 DTHS@    FIXED BIN,         /* POINTER TO 1ST   */
                                                                 /*   ELT OF THIS    */
                                                                 /*   CAT IN THES.   */
                                  03 DLNG     FIXED BIN;         /* LENGTH OF THIS   */
                                                                 /*   CAT IN THES.   */
11                      DCL   01 THSBLOCK CONTROLLED,   /* THESAURUS THESAURUS BLOCK. */
                                02 THSRCD(0:TBLKSIZE-1),
                                  03 TDIR@    FIXED BIN,         /* CHAINING POINTER:*/
                                                                 /*   INDEX IN DRCTRY*/
                                                                 /*   OF NEXT CAT    */
                                                                 /*   CONTAINING THIS*/
                                                                 /*   WORD.          */
                                  03 TVOC@    FIXED BIN;         /* VOCAB POINTER:   */
                                                                 /*   INDEX IN VOCAB */
                                                                 /*   OF THIS WORD.  */
12                      DCL   01 DUMVOCX,               /* BLANK VOCAB RECORD FOR     */
                                02 DUMVOC,              /*   ERASING.                 */
                                  03 DVM  FIXED BIN INITIAL(-1),
                                  03 DVS  FIXED BIN INITIAL(-1),
                                  03 DVD@ FIXED BIN INITIAL(-1),
                                  03 DVC  FIXED BIN INITIAL(-1),
                                  03 DVX  CHAR(1)   INITIAL(' '),
                                  03 DVF  CHAR(1)   INITIAL(' '),
                           03 DVW  CHAR(18)  INITIAL(' ');
 13            DCL   01 VSWAP,                      /* SWAP AREA FOR SORTING */
                            02 VSC            FIXED BIN,
                            02 VSD            FIXED BIN,
                            02 VSE            FIXED BIN,
                            02 VSF            FIXED BIN,
                            02 VSX            CHAR(1),
                            02 VSG            CHAR(1),
                            02 VSH            CHAR(18);
 14            DCL   01 KEY,        /* KEYS TO GENERATE SEARCH-&-PRINTS       */
                            03 KSECT    FIXED BIN,        /* FROM VOCAB        */
                            03 KDIR@    FIXED BIN,        /* FROM VOCAB        */
                            03 KVCOUNT  FIXED BIN,        /* FROM VOCAB        */
                            03 KVFLAG   CHAR(1),          /* FROM VOCAB        */
                            03 KFLAG    CHAR(1),
                            03 KMODE    CHAR(1),          /* FROM REQUEST      */
                            03 KTYPE    CHAR(1),          /* FROM REQUEST      */
                            03 KCOUNT   FIXED BIN,        /* FROM REQUEST      */
                            03 KVOC@    FIXED BIN,
                            03 KDEPTH   FIXED BIN,        /* FROM REQUEST      */
                            03 KRQ#     FIXED BIN,        /* FROM REQUEST      */
                            03 KCAT     CHAR(8),          /* FROM REQUEST      */
                            03 KMATCNT  FIXED BIN,        /* FROM VOCAB        */
                            03 KWORD    CHAR(18);         /* FROM VOCAB        */
 15            DCL   01 TEXTRCD,                     /* TEXT RECORD            */
                            02 TWDLNG       FIXED DEC(2),       /* WORD LENGTH */
                            02 TMATCNT      FIXED BIN,          /* MATCH COUNT */
                            02 TCOUNT       FIXED BIN,          /* WORD COUNT  */
                            02 TWORD        CHAR(18) VARYING;   /* WORD        */
 16            DCL   FILL CHAR(70) VARYING;
 17            DCL   01 NEWTWORDS(0:300),            /* TABLE OF TEXT WORDS    */
                           03 NMATCNT  FIXED BIN,
                           03 NSECT    FIXED BIN,
                           03 NDIR@    FIXED BIN,
                           03 NCOUNT   FIXED BIN,
                           03 NX       CHAR(1),
                           03 NFLAG    CHAR(1),
                           03 NWORD    CHAR(18);
 18            NMATCNT(0) = -3; NWORD(0) = ' ';           /* INITIAL VALUES    */
 20            NCOUNT(0) = 0;
 21            DCL   TTRIPL  CHAR(3) INITIAL(' '), /* CURRENT LEADING TRIPLE*/
                     TPAIR   CHAR(2) INITIAL(' '); /* CURRENT LEADING PAIR  */
 22            DCL   01 T2TBL(300),                  /* AREAS USED IN PROCES- */
                            02 T2MATCNT     FIXED BIN,      /*SING TYPE 2 RQS  */
                            02 T2COUNT      FIXED BIN;      /*                 */
 23            DCL                              /* "STEM" PARAMETERS.         */
                     SAME_ROOT FIXED DEC(1)    /* RETURNED 1 IF BOTH WORDS   */
                               INITIAL (0),    /* HAVE SAME ROOT; OTHERWISE 0*/
                     WDM       CHAR(58) VARYING,
                     WDN       CHAR(58) VARYING,
                     LM        FIXED DEC(2),  /* LENGTH OF WORD M            */
                     LN        FIXED DEC(2);  /* LENGTH OF WORD N            */
 24            DCL   KEYCNT FIXED BIN INITIAL(0);  /* RUNNING TOTAL OF KEYS */
 25                DCL   VRWSW FIXED BIN INITIAL(0);    /* VOCAB REWRITE SWITCH. */
                                                        /* 1 WHEN VOCBLOCK IN    */
                         /* CORE HAS HAD NEW INFORMATION INSERTED.  USED BY      */
                         /* "RWRVOC" SUBRTN TO DETERMINE WHEN REWRITING IS NEC.  */
 26                DCL   VFLAGSAV CHAR(1);          /* OVERFLOW FLAG SAVE AREA.   */
 27                DCL   SUMCOUNTS FIXED BIN(31); /* FOR SUMMING MATCNTS.         */
 28                DCL   MSGCNT     FIXED BIN(31)   /* FOR COUNTING # OF WARNING  */
                                     INITIAL(0);    /* MESSAGES WRITTEN           */
 29                DCL PRT#         CHAR(14) VARYING;  /* FOR LAST MSSG.          */

              /*********************************************************************/
              /*********************************************************************/

 30           REQUEST:
              /*********************************************************************/
              /* THIS SECTION READS IN ALL SEARCH REQUESTS; EDITS THEM FOR CORRECT-*/
              /* NESS. INCORRECT REQUESTS ARE REJECTED.                            */
              /*                                                                   */
              /* TO AVOID MULTIPLE PROCESSINGS OF THE MAIN PROGRAMS, THIS PROCEDURE*/
              /* SHOULD BE RERUN, IF NECESSARY, TO BATCH ANALYSES.                 */
              /* INPUT: TEXTSECT CARD    - MANDATORY. FORMAT:                      */
              /*                               TEXTSECT = N;                       */
              /*                               TEXTSECT = N, MSGPARM = 'NOLIST';   */
              /*                           "N" IS THE NUMBER OF THE SECTION OF TEXT */
              /*                           USED IN THIS ANALYSIS. IT MAY BE ANY POS-*/
              /*                           ITIVE INTEGER.                          */
              /*                           SPECIFYING THE MSG PARAMETER 'NOLIST'   */
              /*                           WILL CAUSE SUPPRESSION OF WARNING       */
              /*                           MESSAGES REGARDING TEXT HANDLING & THE- */
              /*                           SAURUS LINKING.  THE DEFAULT VALUE IS   */
              /*                           'LIST'.                                 */
              /*                           AN OPTIONAL HEADING MAY BE TYPED IN THE */
              /*                           PORTION OF THE TEXTSECT CARD FOLLOWING  */
              /*                           THE ";".  IT WILL BE PRINTED ON THE     */
              /*                           FIRST PAGE OF THE OUTPUT.               */
              /*        ANALYSIS REQUEST CARDS.                                    */
              /*                           PARAMETERS SEPARATED BY COMMAS; LAST    */
              /*                           PARAMETER FOLLOWED BY SEMICOLON.        */
              /*        "GO AHEAD" CARD -  OPTIONAL; THE ANALYSIS WILL PROCEED ONLY */
              /*                           IF THIS CARD IS ENTERED. IT MUST IMMEDI- */
              /*                           ATELY FOLLOW THE ANALYSIS REQUEST CARDS. */
              /*        "RESTART" CARD  -  IF THIS CARD IS ENTERED IN LIEU OF THE  */
              /*                           "GO AHEAD" CARD, THIS PROC WILL RESTART */
              /*                           AT THE BEGINNING, LOOKING FOR MORE      */
              /*                           REQUESTS.                               */
              /*********************************************************************/
                 PUT PAGE;
 31              DCL                                  /* TEXTSECT CARD ENTRIES:    */
                    TEXTSECT     FIXED BIN INITIAL(-1),    /* CURRENT SECTION  */
                                                           /* OF TEXT.         */
                    MSGPARM      CHAR(6) VARYING           /* MSG PARAMETER:   */
                      INITIAL('LIST');                     /* LIST/NOLIST.     */
                                                           /* LIST IS DEFAULT. */
```

```
32                 DCL CH      CHAR(1),                          /* WORKING CHAR      */
                       CH7     CHAR(7),                          /* WORKING CHARS     */
                       CH6     CHAR(6),                          /*                   */
                       RQTX    CHAR(1),                          /* RQ# CHAR          */
                       CARD    CHAR(81);                         /* INPUT CARD AREA   */
33                 DCL RQ#IN             CHAR(20) VARYING,       /* PARAMETERS FROM   */
                       KEYLIST           CHAR(4) VARYING,        /* REQUEST CARD.     */
                        DEPTH             CHAR(2) VARYING,
                       TYPE              CHAR(1),
                       WORD              CHAR(18) VARYING,
                       CAT               CHAR(8), VARYING,
                       THRESHOLD         CHAR(20) VARYING,
                       MODE              CHAR(1);
34                 DCL   01 REQUESTS(100),                       /* REQUEST TABLE     */
                             02 RQ#          FIXED BIN,
                             02 RQKEYL       CHAR(4),
                             02 RQTYPE       CHAR(1),
                             02 RQMODE       CHAR(1),
                             02 RQCAT        CHAR(8),
                             02 RQCOUNT      FIXED BIN,
                             02 RQDEPTH      FIXED BIN,
                             02 RQWORD       CHAR(18) VARYING;
35                 DCL   01 RQSWAP,                              /* SWAP SPACE FOR    */
                             02 RQXA         FIXED BIN,      /*   SORT            */
                             02 RQKG         CHAR(4),
                             02 RQXB         CHAR(1),
                             02 RQXC         CHAR(1),
                             02 RQXD         CHAR(8),
                             02 RQXE         FIXED BIN,
                             02 RQXF         FIXED BIN,
                             02 RQXG         CHAR(18) VARYING;
36          FIRSTCARD: GET EDIT(CARD) (A(80)); SUBSTR(CARD,81,1) = ';';
38                 GET STRING (CARD) DATA;
39                 IF TEXTSECT = -1 THEN
40                       DO;  PUT EDIT ('FIRST CARD DOES NOT GIVE TEXT SECTION.',
                                        ' JOB TERMINATED.') (SKIP,A,A);
42                            GO TO ENDTHESAUR;
43                       END;
44                 DO I = 1 TO 80;
45                       IF SUBSTR(CARD,I,1) = ';' THEN GO TO HDG;
47                 END;
48                 GO TO RQHDG;
49          HDG:   PUT EDIT (SUBSTR(CARD,I+1,80-I)) (A(80-I)); PUT SKIP(2);
51          RQHDG: PUT EDIT('TEXT SECTION',TEXTSECT) (A,F(3));
52                 PUT EDIT('MSGPARM IS ','''',MSGPARM,'''') (X(5),4 A);
53                 PUT EDIT('***** REQUEST EDIT') (SKIP(2),A);
54          RQSTART:ON ENDFILE(SYSIN) GO TO ENDTHESAUR;
56                 IRQMAX = 0;
57          GETRQ: GET EDIT(CH) (A(1)); IF CH = ' ' THEN GO TO GETRQ;
60          GETRQX: IF CH = 'G'  THEN GO TO CHECKGOAHEAD;
62                 IF CH = 'R'  THEN GO TO CHECKRESTART;
64                 IF CH ¬= 'A' THEN DO; REJECT: PUT EDIT
                                     ('REQUEST DOES NOT BEGIN WITH "ANALYSIS"',
                                ' - REJECTED.') (SKIP,A,A);
                               GO TO GETRQ;
                           END;
         GET EDIT  (CH)(A(1)); IF CH ¬= 'N' THEN GO TC REJECT;
  GET3:  GET EDIT  (CH)(A(1)); IF CH ¬= ' ' THEN GO TO GET3;
         RQ#IN = '';
  GET4:  GET EDIT  (CH)(A(1)); IF CH = ' ' THEN GO TO GET4;
         IF CH < '0' THEN GO TO CONVERTRQ#;
  BLDRQ#:RQ#IN = RQ#IN || CH;
         GET EDIT  (CH)(A(1)); IF CH < '0' THEN GO TC CONVERTRQ#;
         GO TO BLDRQ#;
  CONVERTRQ#:
         IF IRQMAX=100 THEN DO; GET DATA; GO TO GETRQ; END;
         IRQMAX = IRQMAX + 1;
         RQ#(IRQMAX)= RQ#IN;
  SETUPRQ:
         TYPE, MODE, WORD, CAT, THRESHOLD, KEYLIST, DEPTH = ' ';
         GET DATA;
         IF CAT ¬= ' ' THEN DO WHILE (LENGTH(CAT)<8);  /* PAD CAT ON   */
                               CAT = ' ' || CAT,       /* LEFT WITH    */
                           END;                        /* BLANKS.      */
         IF MODE=' ' THEN
               DO;  MCDE = 'A';
                    PUT EDIT ('ANALYSIS ',RQ#IN,
                              ' - MODE NOT SPECIFIED. A INSERTED.')
                             (SKIP,A,A,A);
               END:
         IF (TYPE<'1')|(TYPE>'4') THEN
           DO; PUT EDIT('ANALYSIS ',RQ#IN,
                          ' - INCORRECT TYPE. REQUEST REJECTED.')
                         (SKIP,A,A,A);
               GO TO GETRQ;
           END;
         RQKEYL   (IRQMAX) = KEYLIST;    /* ENTER THIS REQUEST IN THE  */
         RQTYPE   (IRQMAX) = TYPE;      -/* REQUEST TABLE.             */
         RQWORD   (IRQMAX) = WORD;
         RQCAT    (IRQMAX) = CAT;
         RQMODE   (IRQMAX) = MODE;
         IF THRESHOLD = ' '
           THEN RQCCUNT(IRQMAX) = 0;
           ELSE RQCOUNT(IRQMAX) = THRESHOLD;
         IF DEPTH = ' '
           THEN RQDEPTH(IRQMAX) = -1;
           ELSE RQDEPTH(IRQMAX) = DEPTH;
         IF RQDEPTH(IRQMAX) > 9 THEN
           DO; RQDEPTH(IRQMAX) = 9;
               PUT EDIT('ANALYSIS ', RQ#IN,
                         ' - MAXIMUM DEPTH PERMITTED IS 9. ',
                         'DEPTH REDUCED TO 9.')
                        (SKIP,A,A,A,A);
           END;
        GO TO GETRQ;                     /* GO GET NEXT REQUEST        */
CHECKGOAHEAD:
```

```
                        GET EDIT (CH7)(A(7));
 127                    IF CH7 = 'O AHEAD'
 128                       THEN DO;  GET EDIT(CH)(X(71),A(1)); GO TO SORTRQS; END;
 132                       ELSE GO TO REJECT;
 133                CHECKRESTART:
                        GET EDIT (CH6)(A(6));
 134                    IF CH6 = 'RSTART' THEN GO TO RQSTART; ELSE GO TO REJECT;
                    /*****************************************************************/
                    /* RANKING SORT OF REQUEST TABLE ON RQTYPE.                      */
 137                SORTRQS:
                        DO I = 1 TO IRQMAX-1;
 138                      IF RQTYPE(I) > RQTYPE(I+1) THEN
 139                        LO: DO J = I+1 TO 2 BY -1 WHILE(RQTYPE(J) < RQTYPE(J-1));
 141                               RQSWAP = REQUESTS(J);
 142                               REQUESTS(J) = REQUESTS(J-1);
 143                               REQUESTS(J-1) = RQSWAP;
 144                             END;
 145                        END;
 146                    END;
 147                    PUT EDIT('***SORTED ANALYSIS REQUESTS TO BE PROCESSED***')
                                (SKIP(2),A);
 148                    PUT EDIT('ANALYSIS #  TYPE MODE THRESHOLD DEPTH KEYLIST ',
                                 'CATEGORY WORD')
                                (SKIP(2),X(14),A,A);
 149                    DO IX = 1 TO IRQMAX;
 150                      PUT EDIT(IX,'.',RQ#(IX),RQTYPE(IX),RQMODE(IX),RQCOUNT(IX),
                                   RQDEPTH(IX),RQKEYL(IX),RQCAT(IX),RQWORD(IX))
                                  (SKIP(2),F(3),A,F(20),X(4),A,X(4),A,F(8),F(8),X(4),
                                   A,X(2),A,X(1),A);
 151                    END;
 152                    PUT PAGE;
                    /*****************************************************************/
 153                TYPESEARCH:
                    /* HERE WE SEE IF THERE ARE ANY RQS OF TYPE 1 OR 2.   IF SO WE POINT */
                    /* IXF&IXL(X=1,2)    TO THE FIRST & LAST OF THEIR RESPECTIVE TYPES.  */
                        I1F,I1L,I2F,I2L =0;
 154                    DO I = 1 TO IRQMAX;
 155                          IF RQTYPE(I) = '1'
 156                             THEN IF I1F = 0
 157                                        THEN I1F,I1L = I;
 158                                        ELSE I1L = I;
 159                             ELSE
 159                          IF RQTYPE(I) = '2'
 160                             THEN IF I2F = 0
 161                                        THEN I2F,I2L = I;
 162                                        ELSE I2L = I;
 163                    END;

                    /*****************************************************************/
                    /*****************************************************************/

 164                INITIALIZE_VOCAB:
                    /*****************************************************************/
```

```
                  /* SCAN THE VOCAB UNDOING THINGS DONE BY PREVIOUS RUNS.  INITIALIZE  */
                  /* THINGS AS FOLLOWS:                                                 */
                  /* (IN GENERAL, THE INITIAL VALUE OF FIELDS IS AS FOLLOWS:            */
                  /*          NUMERIC:     -1                                           */
                  /*          CHARACTER:  BLANKS.)                                      */
                  /*   1.  SET ALL MATCNT & COUNT FIELDS TO -1.                         */
                  /*   2.  ERASE ALL IRRELEVANT TEMPORARY ENTRIES -- I.E., THOSE WITH   */
                  /*       VFLAG = '3' & VSECT >= TEXTSECT.                             */
                  /*   3.  BLANK OUT FLAGS THAT ARE < '3', I.E., '1' OR '2'.            */
                  /*   4.  SET TO -1 ALL VSECTS THAT ARE >= TEXTSECT.                   */
                  /*********************************************************************/
                          PUT EDIT('***** VOCABULARY INITIALIZATION') (A);
  165                     GET FILE(THSCTL)DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE,
                                     VLASTBLK,CLASTBLK,TLASTBLK);
  166                     ALLOCATE VOCBLOCK;
  167                     DO VKEYR = 0 TO VLASTBLK;
  168                       CALL RWRVOC:  VRWSW = 1;
  170                       J = -1;
  171             VSCAN1:   DO IVOC = 0 TO VBLKSIZE-1;
  172                         IF VFLAG(IVOC) = '3' & VSECT(IVOC) >= TEXTSECT THEN
  173                           DO; VOCRCD(IVOC) = DUMVOC;          /* ERASE ENTRY      */
  175                               IF J = -1 THEN J = IVOC; /* REMEMBER 1ST RCD ERASED*/
  177                               GO TO VSCAN1X;
  178                           END;
  179                         VMATCNT(IVOC) = -1;
  180                         VCOUNT(IVOC) = -1;
  181                         IF VFLAG(IVOC) < '3' THEN VFLAG(IVOC) = ' ';
  183                         IF VSECT(IVOC) >= TEXTSECT THEN VSECT(IVOC) = -1;
  185             VSCAN1X:  END;
  186                       IF J > -1 THEN                  /* GARBAGE COLLECTION       */
  187                         DO; VFLAGSAV = VFLAG(VBLKSIZE-1);     /* SAVE OVFLW FLAG  */
  189                             DO I = (J+1) TO (VBLKSIZE-1);
  190                               IF VWORD(I) ¬= ' ' THEN
  191                                 DO; VOCRCD(J) = VOCRCD(I);
  193                                     VOCRCD(I) = DUMVOC;
  194                                     J = J + 1;
  195                                 END;
  196                             END;
  197                             IF VFLAGSAV = '5'
  198                               THEN VFLAG(VBLKSIZE-1) = VFLAGSAV;
  199                         END;
  200                     END;

                  /*********************************************************************/
                  /*********************************************************************/

  201             UPDATE_VOCAB:
                  /*********************************************************************/
                  /* HERE WE UPDATE THE VOCABULARY WITH THE CURRENT SECTION OF TEXT.   */
                  /* SCAN SEQUENTIALLY THE PERMANENT VOCAB AND TEXT, MAKING VOCAB      */
                  /* ENTRIES AS FOLLOWS:                                                */
                  /*      IF TEXT WORD IS IN VOCAB, ENTER MATCNT & COUNT. ALSO ENTER   */
                  /*            SECT IF THIS IS THE FIRST APPEARANCE OF THE WORD.      */
```

```
/*        IF TEXT WORD IS NOT IN VOCAB, ENTER IT IN NEWTWORDS TABLE.    */
/*        WHEN THERE IS A BREAK ON THE LEADING TRIPLE IN THE TEXT, CALL*/
/*              THE "TRIPLBRK" SUBRTN.  THERE MERGE NEWTWORDS WITH TEMP*/
/*              ORARY ENTRIES ALREADY IN VOCAB, INSERT ALL NEW TEMP     */
/*              ENTRIES (VFLAG='3'), & FLAG PERMANENT ENTRIES THAT      */
/*              HAVE THE SAME MATCH#.  IF THERE ARE NO PERMANENT        */
/*              ENTRIES WITH SAME MATCH#, PRODUCE MSSG & MAKE NO ENTRY.*/
/***********************************************************************/
          PUT EDIT('***** THESAURUS UPDATE & SEARCH KEY GENERATION')
                   (SKIP(2),A);
KIS:      KSECT = TEXTSECT;                   /* WRITE A DUMMY KEY WITH      */
          KRC# = -1000;                       /* CURRENT TEXTSECT, TO BE 1ST*/
          KCAT = ' '; KWDEF = ' ';            /* KEY READ BY "SEPRINT".      */
          KMATCNT = 0;
          WRITE FILE(CURKEYS)FROM(KEY);
          ON ENDFILE(TEXT)
            BEGIN; TWORD=(18)'9'; CALL TRIPLBRK; GO TO UPDVOCEND; END;
          GET FILE(THSCTL)EDIT(VKEY,VLEXT)(F(4));     /* GET VOCAB BUCKET */
                                                      /*    KEYS.          */
          GET FILE(TEXT)EDIT(TWDLNG,TMATCNT,TCOUNT,TWORD,FILL)
                              (F(2),F(5),F(5),A(TWDLNG),A(18-TWDLNG));
          TTRIPL = SUBSTR(TWORD,1,3);
          IT2T = 0; I2TBL = -1; INEWT = 0;
          GO TO GETVKEY;
GETT:     GET FILE(TEXT)EDIT(TWDLNG,TMATCNT,TCOUNT,TWORD,FILL)
                              (F(2),F(5),F(5),A(TWDLNG),A(18-TWDLNG));
          IF TTRIPL < SUBSTR(TWORD,1,3) THEN CALL TRIPLBRK;
GETVKEY:IF TPAIR = SUBSTR(TWORD,1,2) THEN GO TO SUMT2;
          TPAIR = SUBSTR(TWORD,1,2);
          IVK = UNSPEC(SUBSTR(TPAIR,1,1));      /* COMPUTE VKEY INDICES  */
          JVK = UNSPEC(SUBSTR(TPAIR,2,1));
          IF IVK < 202 THEN IVK = IVK - 192;
            ELSE IF IVK < 218 THEN IVK = IVK - 199;
              ELSE IVK = IVK - 207;
          IF JVK < 202 THEN JVK = JVK - 192;
            ELSE IF JVK < 218 THEN JVK = JVK - 199;
              ELSE JVK = JVK - 207;
          IF (IVK<1)|(IVK>26)|(JVK<1)|(JVK>26) THEN
            DO; DO WHILE(TPAIR=SUBSTR(TWORD,1,2));
                  IF MSGPARM = 'LIST' THEN
                    PUT EDIT('TEXT WORD ',TWORD,' SKIPPED. COUNT =',
                             TCOUNT)
                             (SKIP,A,A,COLUMN(29),A,F(7));
                  MSGCNT = MSGCNT+1;
                  GET FILE(TEXT)EDIT(TWDLNG,TMATCNT,TCOUNT,TWORD,FILL)
                              (F(2),F(5),F(5),A(TWDLNG),A(18-TWDLNG));
                END;
                GO TO GETVKEY;
            END;
VFETCH: IF VKEY(IVK,JVK) = VLASTBLK THEN GO TO SUMT2;
          VKEYR = VKEY(IVK,JVK);
          CALL RWRVOC;
          IVOC = 0;
```

```
SUMT2:
   /* IF THERE ARE TYPE2 REQUESTS, SUM TOKENS IN MATCNT IN T2TBL.      */
          IF I2F = 0 THEN GO TO CMPTV;
          DO I = 1 TO IT2T;
            IF TMATCNT = T2MATCNT(I) THEN
              DO; T2COUNT(I) = T2COUNT(I) + TCOUNT;
                   GO TO CMPTV;
              END;
          END;
          IT2T = I;
          T2MATCNT(IT2T) = TMATCNT;
          T2COUNT(IT2T) = TCOUNT;
CMPTV:    IF VKEY(IVK,JVK) = VLASTBLK THEN GO TO TVNOTFOUND;
          IF VWORD(IVOC) = ' ' THEN GO TO TVNOTFOUND;
          IF VFLAG(IVOC) > '2' THEN GO TO TVNOTFOUND;
          IF VWORD(IVOC) > TWORD THEN GO TO TVNOTFOUND;
          IF VWORD(IVOC) = TWORD THEN GO TO TVFOUND;
          IVOC = IVOC + 1;
          IF IVOC < VBLKSIZE THEN GO TO CMPTV;
          /* END OF BLOCK.  IF THERE IS ANOTHER BLOCK, GO GET IT;       */
          /* OTHERWISE, NOT FOUND.                                      */
          IF VINCOREKEY < (VKEY(IVK,JVK)+VEXT(IVK,JVK)) THEN
            DO; VKEYR = VINCOREKEY+1;
                 CALL RWRVOC;
                 IVCC = 0;
                 GO TO CMPTV;
            END;
TVNOTFOUND:
          INEWT = INEWT + 1;
          NEWTWORDS(INEWT). = DUMVOC;
          NMATCNT(INEWT) = TMATCNT;
          NCOUNT(INEWT) = TCOUNT;
          NWORD(INEWT) = TWORD;
          NSECT(INEWT) = TEXISECT;
          GO TO GETT;
TVFOUND:
          VRWSW = 1;
          VMATCNT(IVOC) = TMATCNT;
          VCOUNT(IVOC) = TCOUNT;
          IF VSECT(IVOC) = -1 THEN VSECT(IVOC) = TEXTSECT;
          GO TO GETT;
UPDVOCEND: IF VRWSW = 1 THEN
               DO; VRWSW = 0;
                    REWRITE FILE(VOCAB) FROM(VOCBLOCK) KEY(VINCOREKEY);
               END;

/*********************************************************************/
/*********************************************************************/

BUILD_KEYS:
/*********************************************************************/
/* HERE WE PROCESS THE REQUEST TABLE TO FINISH THE TABLE OF KEYS.    */
/* TYPE 2 REQUESTS WERE PROCESSED IN "PAIRBRK" SECTION OF "TRIPLBRK" */
/* SUBROUTINE CALLED BY "UPDATE_VOCAB".                              */
/*********************************************************************/
        PUT EDIT('***** THESAURUS UPDATE COMPLETE')(SKIP(2),A);
        PUT SKIP;
        ALLOCATE DIRBLOCK, THSBLOCK;
        IRQ = 0;                              /* INITIALIZE REQUEST INDEX   */
NEXTRQ: IF IRQ = IRQMAX THEN GO TO KEYS_COMPLETE;
        IRQ = IRQ + 1;
        RQTX = RQTYPE(IRQ);
        IF RQTX = '1' THEN GO TO TYPE1;
        IF RQTX = '3' THEN GO TO TYPE3;
        IF RQTX = '4' THEN GO TO TYPE4;
        GO TO NEXTRQ;
TYPE1:
/* KEY ON EVERY CATEGORY THAT APPEARS MORE THAN RQCOUNT TIMES IN     */
/* TEXT:                                                             */
/* FOR EACH CATEGORY, WE TOTAL THE COUNTS OF ALL WORDS IN THE CAT.   */
/* (FLAG'3' ENTRIES ARE NOT YET INCLUDED IN COUNT.)                  */
/* EACH TYPE 1 REQUEST HAS A THRESHOLD, RQCOUNT. FOR EACH ONE, IF THE*/
/* TOTAL OF COUNTS >= RQCOUNT, WE GENERATE A KEY FOR THE CATEGORY.   */
        DO DKEYP = 0 TO DLASTBLK;             /* SEQUENTIAL SCAN OF DECTRY  */
          CALL READDIR;
DSCAN1:   DO IDIR = 0 TO DBLKSIZE-1;
            SUMCOUNTS = 0;
CATSCAN:    TKEYP = DTHS@(IDIR) / TBLKSIZE;
            ITHS = DTHS@(IDIR) - (TKEYP*TBLKSIZE);
            CALL READTHS;
            ITHS = ITHS - 1;
TSCAN1:     DO J = 1 TO DLNG(IDIR);           /* SCAN CATEGORY IN THES.     */
              ITHS = ITHS + 1;
              IF ITHS = TBLKSIZE THEN
                DO; TKEYP = TKEYP + 1;
                    CALL READTHS;
                    ITHS = 0;
                END;
              VKEYP = TVOC@(ITHS) / VBLKSIZE;
              IVOC = TVOC@(ITHS) - (VKEYP*VBLKSIZE);
              CALL RWPVOC;
              IF VCOUNT(IVOC) > 0 THEN
                SUMCOUNTS = SUMCOUNTS+VCOUNT(IVOC);
TSCAN1X:    END;
ENTERKEYS1: DO IRQ = I1F TO I1L;              /* HERE WE HAVE SUMMED THE    */
              IF SUMCOUNTS                    /* TOKENS IN THE CAT.  NOW    */
                >= RQCOUNT(IRQ) THEN          /* SCAN TYPE 1 REQUESTS.  FOR */
                DO; KRQ# = RQ#(IRQ);          /* EACH, IF SUMCOUNTS MEETS   */
                    KCAT = DCAT(IDIR);        /* THE THRESHOLD, GENERATE A  */
                    KWORD = ' ';              /* KEY FOR THE CAT.           */
                    KMODE = RQMODE(IRQ);
                    KTYPE = '1';
                    KCOUNT = RQCOUNT(IRQ);
                    KDEPTH = RQDEPTH(IRQ);
                    KVOC@ = -1;
                    KMATCNT = 0;
                    KSECT = -1;
                    KDIR@ = (DKEYR*DBLKSIZE)+IDIR;
                    KVCOUNT = SUMCOUNTS;
                    KVFLAG = ' ';
                    KFLAG = ' ';
                    WRITE FILE(CURKEYS)FROM(KEY) ;
                    KEYCNT = KEYCNT + 1;
                    IF ROKEYL(IRO) = 'LIST' THEN CALL KEYPRINT;
                END;
            END;
ESCAN1X:  END;
        END;
    /* HERE WE HAVE GONE THRU THE WHOLE DIRECTORY. WE SET THE REQUEST*/
    /* TABLE INDEX TO SHOW THAT WE HAVE PROCESSED ALL TYPE 1 REQUESTS*/
    /* AND THEN RETURN.                                              */
        IRO = I1L;
        GO TO NEXTRO;
TYPE3:
/* KEY ON EACH WORD IN ROCAT:                                        */
/* SCAN DRCTRY TO FIND CAT.  GENERATE A KEY FOR EACH WORD IN CAT.    */
/* INCLUDE TEMP (VFLAG = '3') ENTRIES WITH SAME MATCNT.              */
        DO DKEYR = 0 TO DLASTBLK;
          CALL READDIR;
          DO IDIR = 0 TO DBLKSIZE-1;
            IF ROCAT(IRO) = DCAT(IDIR) THEN GO TO KEY3MODE;
          END;
        END;
  /* ROCAT NOT IN THESAURUS.  WRITE A MSSG & REJECT THE REQUEST.     */
        PUT EDIT('ANALYSIS ',RO#(IRO),' - CATEGORY NOT IN THESAURUS. ',
                 'REJECTED.') (SKIP,A,F(5),A,A);
        GO TO NEXTRO;
KEY3MODE:
        IF ROMODE(IRO) > 'C' THEN GO TO ENTERKEY3;
                                       /* HERE WE MUST SEE IF THE CAT*/
                                       /* IS IN THE TEXT.            */
        TKEYR = DTHS@(IDIR) / TBLKSIZE;     /* FIND CAT IN THES.     */
        ITHS = DTHS@(IDIR) - (TKEYR*TBLKSIZE);
        CALL READTHS;
        ITHS = ITHS - 1;
TSCAN2: DO J = 1 TO DLNG(IDIR);            /* SCAN CAT               */
          ITHS = ITHS + 1;
          IF ITHS = TBLKSIZE THEN
            DO; TKEYR = TKEYR + 1;
                CALL READTHS;
                ITHS = 0;
            END;
          VKEYR = TVOC@(ITHS) / VBLKSIZE;   /*    CH VOCAB ENTRY     */
          IVOC = TVOC@(ITHS) - (VKEYR*VBL  IZE);
          CALL RWRVOC;
          IF VMATCNT(IVOC) > -1 THEN     /* HERE WE FOUND A WORD IN  */
            GO TO ENTERKEY3;             /* TEXT. GO ENTER THE KEY.  */
TSCAN2X:END;
        GO TO NEXTRO;
```

```
       ENTERKEY3.                                          /* GENERATE A KEY.  */
               KRQ# = RQ#(IRQ);
               KCAT = RQCAT(IRQ);
               KWORD = ' ';
               KMODE = RQMODE(IRQ);
               KTYPE = '3';
               KCOUNT = RQCOUNT(IRQ);
               KDEPTH = RQDEPTH(IRQ);
               KVOC@ = -1;
               KMATCNT = 0;
               KSECT = -1;
               KDIR@ = (DKEYR*DBLKSIZE)+IDIR;
               KVCOUNT = -1;
               KVFLAG = ' ';
               KFLAG = ' ';
               WRITE FILE(CURKEYS)FROM(KEY);
               KEYCNT = KEYCNT + 1;
               IF RQKFYL(IRQ) = 'LIST' THEN CALL KEYPRINT;
               GO TO NEXTRQ;
       TYPE4:
       /* KEY ON RQWORD:                                                       */
       /* FIND VOCAB ENTRY FOR RQWORD & GENERATE ONE KEY FROM IT.  IF THERE */
       /* IS NO ENTRY, WRITE MSSG AND REJECT REQUEST.                          */
       /* THE WORD MUST BE EITHER IN THE ORIGINAL THESAURUS (VFLAG<'3') OR    */
       /* IN THE CURRENT SECTION OF TEXT (VMATCNT>-1).  IF NOT, WRITE MSSG    */
       /* AND REJECT REQUEST.                                                  */
               IVK = UNSPEC(SUBSTR(RQWORD(IRQ),1,1));   /* COMPUTE VKEY      */
               JVK = UNSPEC(SUBSTR(RQWORD(IRQ),2,1));   /* INDICES.          */
               IF IVK < 202 THEN IVK = IVK - 192;
                 ELSE IF IVK < 218 THEN IVK = IVK - 199;
                   ELSE IVK = IVK - 207;
               IF JVK < 202 THEN JVK = JVK - 192;
                 ELSE IF JVK < 218 THEN JVK = JVK - 199;
                   ELSE JVK = JVK - 207;
               VKEYR = VKEY(IVK,JVK);           /* FETCH VOCAB BUCKET & SCAN  */
               CALL RWRVOC;                              /* FOR RQWORD.       */
       VSCAN5: DO IVOC = 0 TO VBLKSIZE WHILE(VWORD(IVOC) ¬= ' ');
                 IF VWORD(IVOC) = RQWORD(IRQ) THEN GO TO KEY4FOUND;
               END;
               IF VINCOREKEY < VKEY(IVK,JVK)+VEXT(IVK,JVK) THEN
                 DO; VKEYR = VINCOREKEY + 1;  /* IF THERE ARE SUBSEQUENT     */
                     CALL READVOC;            /* BLOCKS, GET THE NEXT &      */
                     GO TO VSCAN5;            /* CONTINUE.                   */
                 END;
               IF VFLAG(VBLKSIZE-1) = '5' THEN      /* CHECK OVERFLOW BUCKET */
                 DO; VKEYR = VLASTBLK;.
                     CALL READVOC;
                     GO TO VSCAN5;
                 END;
          /* AT THIS POINT, NOT FOUND.  WRITE A MSSG & REJECT THE REQUEST.   */
               PUT EDIT('ANALYSIS ',RQ#(IRQ),' - WORD IN NEITHER THESAURUS ',
                          'NOR TEXT. REJECTED.')(SKIP,A,F(5),A,A);
               GO TO NEXTRQ;
          KEY4FOUND:
            /* IF THIS IS A TEMPORARY ENTRY, THEN IT MUST BE IN THE CURRENT     */
            /* SECTION OF TEXT.  OTHERWISE, REJECT.                             */
                  IF VFLAG(IVOC) > '2' & VMATCNT(IVOC) = -1 THEN
                    DO: PUT EDIT('ANALYSIS ',RQ#(IRQ),' - WORD IN NEITHER ',
                                 'ORIGINAL THESAURUS NOR CURRENT SECTION OF ',
                                 'TEXT. REJECTED.')(SKIP,A,F(5),A,A,A);
                        GO TO NEXTRQ;
                    END;
          ENTERKEY4:
                  KRQ# = RQ#(IRQ);                              /* GENERATE A KEY.  */
                  KCAT = RQCAT(IRQ);
                  KWORD = RQWORD(IRQ);
                  KMATCNT = VMATCNT(IVOC);
                  KSECT = VSECT(IVOC);
                  KDIR# = VDIR#(IVOC);
                  KVCOUNT = VCOUNT(IVOC);
                  KVFLAG = VFLAG(IVOC);
                  KVOC# = (VINCOREKEY*VFILESIZE) + IVOC;
                  KFLAG = ' ';
                  KMODE = RQMODE(IRQ);
                  KTYPE = RQTYPE(IRQ);
                  KCOUNT = RQCOUNT(IRQ);
                  KDEPTH = RQDEPTH(IRQ);
                  WRITE FILE(CURKEYS)FROM(KEY);
                  KEYCNT = KEYCNT + 1;
                  IF RQKEYL(IRQ) = 'LIST' THEN CALL KEYPRINT;
                  GO TO NEXTRQ;
          /*********************************************************************/
          KEYS_COMPLETE: GO TO ENDTHESAUR;

          /*********************************************************************/
          /*********************************************************************/
          /*********                                ****************************/
          /*********     S U B R O U T I N E S      ****************************/
          /*********                                ****************************/
          /*********************************************************************/
          /*********************************************************************/

          RWRVOC: PROC;
          /*********************************************************************/
          /* THIS SUBROUTINE IS USED FOR ALL VOCAB FETCHES.                    */
          /* THE KEY OF THE REQUESTED BLOCK MUST BE IN VKEYR.  BEFORE READING, */
          /*   THIS IS COMPARED WITH VINCOREKEY TO SEE IF THE REQUESTED BLOCK  */
          /*   IS ALREADY IN CORE.  IF SO, READING IS BYPASSED.  AFTER READING */
          /*   A NEW BLOCK, VINCOREKEY IS UPDATED TO THE NEW KEY VALUE.        */
          /* THERE ARE TWO ENTRY POINTS:                                       */
          /*   RWRVOC:  BLOCK IN CORE IS REWRITTEN BEFORE REQUESTED BLOCK IS   */
          /*            READ.                                                  */
          /*   READVOC: REQUESTED BLOCK IS READ , OVERLAYING WHATEVER WAS IN   */
          /*            CORE.                                                  */
          /*********************************************************************/
                  IF VKEYR = VINCOREKEY THEN GO TO RWRVOCEND;
```

```
                IF VRWSW = 1 THEN
                      REWRITE FILE(VOCAB)FROM(VOCBLOCK)KEY(VINCOREKEY);
                GO TO FVCC;
        READVOC: ENTRY;
                IF VKEYR = VINCOREKEY THEN GO TO RWRVOCEND;
        FVOC:   READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEYR);
                VINCOREKEY = VKEYR;
                VRWSW = 0;
        RWRVOCEND: END RWRVOC;

        /**********************************************************************/
        /**********************************************************************/

        READDIR: PROC;
        /**********************************************************************/
        /* READ SUBROUTINE FOR DRCTRY DATA SET.                               */
        /**********************************************************************/
                IF DKEYR ¬= DINCOREKEY THEN
                  DO; READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEYR);
                      DINCOREKEY = DKEYR;
                  END;
                END READDIR;

        /**********************************************************************/
        /**********************************************************************/

        READTHS: PROC;
        /**********************************************************************/
        /* READ SUBROUTINE FOR THES DATA SET.                                 */
        /**********************************************************************/
                IF TKEYR ¬= TINCOREKEY THEN
                  DO; READ FILE(THES)INTO(THSBLOCK)KEY(TKEYR);
                      TINCOREKEY = TKEYR;
                  END;
                END READTHS;

        /**********************************************************************/
        /**********************************************************************/

        KEYPRINT: PROC;
        /**********************************************************************/
        /* ROUTINE FOR PRINTING KEYS LISTING.                                 */
        /**********************************************************************/
                DCL KEYHDGF CHAR(1) STATIC INITIAL(' ');
                IF KEYHDGF = ' ' THEN  /* PRINT A HEADING, IF NECESSARY       */
                  DO; KEYHDGF = '1';
                      PUT EDIT('REQUEST# TYPE MODE THRESHOLD MATCNT DEPTH',
                               '    VOC@    DIR@ SECT VCOUNT VFLAG KFLAG',
                               ' CATEGORY WORD')
                              (SKIP,COLUMN(10),A,A,A);
                      PUT SKIP;
                  END;
                PUT EDIT(KRQ#,KTYPE,KMODE,KCOUNT,KMATCNT,KDEPTH,KVOC@,KDIR@,
                         KSECT,KVCOUNT,KVFLAG,KFLAG,KCAT,KWCRD)
                        (SKIP,F(17),X(3),A,X(4),A,F(11),F(7),F(4),F(10),F(8),
                         F(4),F(8),X(3),A,X(5),A,X(3),A,X(1),A);
                   END KEYPRINT;

            /**********************************************************************/
            /**********************************************************************/

            TRIPLBRK: PROC;
            /**********************************************************************/
            /* THIS SUBROUTINE IS CALLED BY UPDATE_VOCAB WHEN THERE IS A BREAK    */
            /* IN THE LEADING TRIPLE OF THE TEXT WORDS.                            */
            /* HERE WE PROCESS ANY ENTRIES IN NEWTWORDS.  THESE ARE TEXT WORDS    */
            /* NOT IN VOCAB.  FOR EACH NEWTWORD WE MAKE A TEMPORARY VOCAB RECORD */
            /* (VFLAG = '3') AND MARK ALL PERMANENT VOCAB ENTRIES THAT HAVE THE   */
            /* SAME MATCNT BY SETTING THEIR VFLAG TO '1' OR '2' APPROPRIATELY.   */
            /**********************************************************************/
                    VKEYSAVE = VINCOREKEY;  IVOCSAVE = IVOC; /* RESTORE AT END    */
                                                           /*   OF SUBROUTINE. */
                    IF INEWT = 0 THEN GO TO TRIPLBRKEND;     /* NO NEWTWORDS TO   */
                                                           /*   PROCESS.        */
                    INEWTMAX = INEWT;
                    NMATCNT(INEWTMAX+1) = 31000;
              /* FIRST SCAN THE END OF THE BUCKET FOR FLAG3 ENTRIES THAT ARE ALSO*/
              /* IN NEWTWORDS.  IF FOUND, COPY THEM INTO NEWTWORDS AND ERASE IN  */
              /* VOCAB.  THE FLAG '3' ENTRIES WHICH ARE NOT IN NEWTWORDS ARE     */
            /* SAVED IN THE BUCKET, WITH A MATCNT OF -1.                          */
                    VKEYR = VKEY(IVK,JVK) + VEXT(IVK,JVK);   /* GET LAST BLOCK    */
                    CALL RWRVOC;                             /*   IN BUCKET.      */
            VSCAN2: DO IVOC = 0 TO VBLKSIZE-1 WHILE(VWORD(IVOC) ¬= ' ');
                      IF VFLAG(IVOC) < '3' THEN GO TO VSCAN2X;
                      IF VWORD(IVOC) > NWCRD(INEWTMAX) THEN GC TO VSCAN2X;
                      DO INEWT = 1 TO INEWTMAX WHILE(VWORD(IVOC)>NWORD(INEWT));
                      END;
                      IF NWORD(INEWT) = VWORD(IVOC) THEN
                        DO; NSECT(INEWT) = VSECT(IVOC);
                            VOCRCD(IVOC) = DUMVOC;
                            VRWSW = 1;
                        END;
            VSCAN2X:END;
              /* GARBAGE COLLECTION                                                */
                    VFLAGSAV = VFLAG(VBLKSIZE-1);  /* SAVE OVERFLOW FLAG           */
                    DO IVOC = 0 TO VBLKSIZE-1 WHILE(VWORD(IVOC) ¬= ' ');
                    END;
                    DO I = (IVOC+1) TO (VBLKSIZE-1);
                      IF VWORD(I) ¬= ' ' THEN
                        DO; VOCRCD(IVOC) = VOCRCD(I);
                            VOCRCD(I) = DUMVOC;
                            IVOC = IVOC + 1;
                            VRWSW = 1;
                        END;
                    END;
                    IF VFLAGSAV = '5' THEN VFLAG(VBLKSIZE-1) = VFLAGSAV;
```

```
           /* IF NEEDED, SCAN OVERFLOW BLOCK.                                          */
548              IF VFLAG(VBLKSIZE-1) ¬= '5' THEN GO TO NSCFT; /* OVERFLOW             */
                                                          /* BLOCK NEVFF HAS AN OVPL FLAG */
550      CVFLSCAN:
                 VKEYR = VLASTBLK;
551              CALL RWRVOC;
552              GO TC VSCAN2;
         /* (AT THIS POINT, IVOC IS POINTING AT THE FIRST AVAILABLE BLANK             */
         /*  SPACE IN THE BUCKET.  IT IS SOMEWHERE EITHEP IN THE LAST BLOCK           */
         /*  IN THE BUCKET OR IN THE OVERFLOW BUCKET.)                                */
           /* PANKING SCRT OF NEWTWORDS ON MATCNT, WORD.                               */
553      NSORT:
                 DO I = 1 TO INEWTMAX-1;
554                IF NMATCNT(I) > NMATCNT(I+1) THEN
555                  DO; DO J = I+1 TO 2 BY -1 WHILE(NMATCNT(J)<NMATCNT(J-1));
557                        VSWAP = NEWTWORDS(J);
558                        NEWTWORDS(J) = NEWTWORDS(J-1);
559                        NEWTWORDS(J-1) = VSWAP;
560                      END;
561                  END;
562              END;
           /* SCAN THE VOCAB BUCKET.  FOR EACH PERMANENT ENTRY IN VOCAB:               */
           /* IF IT HAS A MATCNT THAT ALSO APPEARS IN NEWTWORDS, SET ITS               */
           /*    VFLAG TO '1'.                                                         */
           /* IF IT DOES NOT HAVE A MATCNT, USE "STEM" TC LOCK FOR A MATCH IN          */
           /*    NEWTWORDS.  IF FOUND, INSERT MATCNT & SET VFLAG TC '2'.               */
           /* IN EITHER CASE, SET NFLAG TO '4' TO SIGNAL THAT A MATCH HAS              */
           /*    BEEN FOUND IN VOCAB.                                                  */
563              IF VKEY(IVK,JVK) = VLASTBLK THEN GO TO TEMPENT;
565              VKEYR = VKEY(IVK,JVK);
566              CALL RWRVOC;
567      VSCAN3: DO IVOC = 0 TO VBLKSIZE-1 WHILE(VWORD(IVCC) ¬= ' ');
568                IF SUBSTR(VWORD(IVOC),1,3) ¬= SUBSTR(NWORD(1),1,3)
569                  THEN GO TO VSCAN3X;
570                IF VMATCNT(IVOC) ¬= -1 THEN
571                  DO; DO INEWT = 1 TO INEWTMAX;   /* LOCK FOR THIS MATCNT  */
573                        IF VMATCNT(IVOC) = NMATCNT(INEWT) THEN
574                          DO; VFLAG(IVOC) = '1';
576                              NFLAG(INEWT) = '4';
577                              VRWSW = 1;
578                              GO TO VSCAN3X;
579                          END;
580                      END;
581                      GO TO VSCAN3X;
582                  END;
583                DO INEWT = 1 TO INEWTMAX;    /* USE STEM TO LOCK FCR ROOT  */
584                  IF NMATCNT(INEWT) ¬= NMATCNT(INEWT-1) THEN      /* MATCHES*/
585                    DO; WDM = NWORD(INEWT);  /* NEED COMPARE WITH ONLY 1ST */
587                        WDN = VWORD(IVOC);   /*   WORD IN EACH MATCNT GROUP*/
588                        I = INDEX(WDM,' ');  /* REMOVE TRAILING BLANKS.    */
589                        IF I > 0 THEN WDM = SUBSTR(WDM,1,I-1);
591                        I = INDEX(WDN,' ');
592                        IF I > 0 THEN WDN = SUBSTR(WDN,1,I-1);
```

```
                   LM = LENGTH(WDM);
                   LN = LENGTH(WDN);
                   CALL STEM(SAME_ROOT,LM,WDM,LN,WDN);
                   IF SAME_ROOT = 1 THEN
                     DO: VMATCNT(IVOC) = NMATCNT(INEWT);
                         VFLAG(IVOC) = '2';
                         VRWSW = 1;
                         NFLAG(INEWT) = '4';
                         GO TO VSCAN3X;
                     END;
              END;
           END;
VSCAN3X:END;
  /* IF THERE ARE SUESEQUENT ELOCKS IN THE BUCKET, FETCH THE NEXT    */
  /*     ONE AND REITERATE.                                              */
        IF VINCOREKEY <- (VKEY(IVK,JVK)+VEXT(IVK,JVK)) THEN
          DO: VKEYR = VINCOREKEY+1;
              CALL RWRVOC;
              GO TO VSCAN3;
          END;
/* (AT THIS POINT, IVCC IS POINTING AT THE FIRST AVAILABIE BLANK      */
/*  SPACE IN THE BUCKET.  IT IS SOMEWHERE IN THE LAST BLOCK IN THE   */
/*  BUCKET.)                                                           */
  /* ALL POSSIBLE LINKING HAS BEEN DONE.  NOW MAKE TEMPORARY ENTRIES */
  /*     (VFLAG = '3') FROM NEWTWORDS INTO VOCAB.                       */
TEMPENT:DO INEWT = 1 TO INPWTMAX;
          DO I = INEWT TO INEWTMAX;
            IF NFLAG(I) = '4' THEN GO TO ENTERMCGP;
            IF NMATCNT(I+1) ¬= NMATCNT(I) THEN GO TO NOLINK;
          END;
NOLINK:                                /* REJECT THIS MATCNT GROUP   */
          DO I = INEWT TO INEWTMAX;    /*   BECAUSE UNABIE TC LINK.  */
            IF MSGPARM = 'LIST' THEN
            PUT EDIT('WORD ',NWORD(I),'IN SECTION',TEXTSECT,
                     ': UNABLE TO ESTABLISH ANY REIATIONSHIPS IN ',
                     'THESAURUS. COUNT =',NCOUNT(I))
                    (SKIP,A,A(18),A,F(3),A,A,F(7));
            MSGCNT = MSGCNT+1;
            IF NMATCNT(I+1) ¬= NMATCNT(I) THEN GO TC NOLINKEND;
          END;
NOLINKEND:INEWT = I;
          GO TO TEMPENTX;
ENTERMCGP:DO I = INEWT TO INEWTMAX;    /* MAKE TEMF ENTRIES                */
            IF IVOC = VBLKSIZE THEN    /* OVERFLOW IF NECESSARY            */
              DO; IF VINCOREKFY = VLASTBLF
                    THEN DO:
CVFLOVFL:                        IF MSGPARM = 'LIST' THEN
                                 FUT EDIT('WORD ',NWORD(I),'IN SECTION',
                                          TEXTSECT,': UNABLE TO ENTER IN ',
                                          'VOCAB BECAUSE OF LACK OF ',
                                          'SPACE. COUNT =',NCOUNT(I))
                                         (SKIP,A,A(18),A,F(3),A,A,A,F(7));
                                 MSGCNT = MSGCNT+1;
```

```
                              GO TO ENTMCGPX;
                         END;
                    ELSE DO;  VFLAG(VBLKSIZE-1) = '5'; /*SET OVFL FLAG*/
                              VKEYR = VLASTBLK;          /* FETCH OVFL  */
                              CALL RWRVOC;               /* BLOCK.      */
                              DO IVOC = 0 TO VBLKSIZE-1 /* FIND 1ST    */
                                WHILE(VWORD(IVOC) ¬= ' '); /* BLANK    */
                              END;                           /* ENTRY    */
                              IF IVOC = VBLKSIZE THEN GO TO OVFLOVFL;
                         END;
              END;
            VRWSW = 1;
            VOCRCD(IVOC) = NEWTWORDS(I);
            VFLAG(IVOC) = '3';
            IVOC = IVOC + 1;
            IF NMATCNT(I+1) ¬= NMATCNT(I) THEN GO TO ENTMCGPEND;
ENTMCGPX: END;
ENTMCGPEND: INEWT = I;
TEMPENTX:
        END;
TRIPLBRKEND: INEWT = 0;                 /* ERASE NEWTWORDS TABLE.     */
        TTRIPL = SUBSTR(TWORD,1,3);     /* INSTALL NEW TRIPLE         */
        IF TPAIR ¬= SUBSTR(TWORD,1,2) THEN GO TO PAIRBRK;
        VKEYR = VKEYSAVE;               /* RESTORE TO SEQUENTIAL POSN */
        IVOC = IVOCSAVE;                /*   IN VOCAB.                */
        CALL RWRVOC;
        RETURN;                         /*   AND RETURN.              */
PAIRBRK:
  /* THERE IS WORK TO DO HERE ONLY IF THERE ARE TYPE2 REQUESTS.       */
        IF I2F = 0 THEN GO TO PAIRBRKEND;
TYPE2:
  /* GENERATE KEYS FOR EACH MATCNT(ROOT) THAT OCCURS MORE THAN        */
  /* RQCOUNT TIMES IN THE TEXT:                                       */
  /* FOR EACH MATCNT SUM THE TOKENS IN THE TEXT.  IF THIS SUM >=      */
  /* RQCOUNT, GENERATE KEYS FOR EACH TYPE IN THE MATCNT.              */
  /* SUMMING IS DONE AS TEXT IS PASSED SEQUENTIALLY IN UPDATE_VOCAB */
  /* SECTION OF MAIN PROGRAM.  SUMS ARE KEPT IN T2TBL, BY MATCNT.     */
  /* AT THE END OF EACH BUCKET (BREAK ON TPAIR) T2TBL IS PASSED      */
  /* AGAINST TYPE2 REQUESTS & KEYS ARE GENERATED FOR EACH WORD IN     */
  /* EACH QUALIFYING MATCNT.                                          */
  /* FIRST, SCAN T2TBL RETAINING ONLY ELIGIBLE MATCNTS.               */
        T2FOUNDSW = 0;
T2SCAN: DO I = 1 TO IT2T;
          DO J = I2F TO I2L;
            IF T2COUNT(I) >= RQCOUNT(J) THEN
              DO; T2FOUNDSW = 1;
                  GO TO T2SCANX;
              END;
          END;
          T2MATCNT(I) = -1;             /* ERASE THIS MATCNT          */
T2SCANX:END;
        IF T2FOUNDSW = 0 THEN GO TO PAIRBRKEND;
        T2FOUNDSW = 0;
```

```
ENTERKEYS2:
  /* SCAN THE BUCKET.  FOR FACH WORD, GENFRATE A KEY FOR EACH PEQUEST*/
  /* WHOSE RQCCUNT IS EQUALLEI OF EXCEEDED BY THE CCRRESFONDING      */
  /* T2COUNT.                                                        */
         VKEYR = VKEY(IVK,JVK);
         CALL RWRVOC;
VSCAN4: DO IVOC = 0 TO VBLKSIZE-1 WHILE(VWCRD(IVCC) ¬= ' ');
           IF VMATCNT(IVOC) = -1 THEN GO TO VSCAN4X;
           DO I = 1 TC IT2T;
              IF VMATCNT(IVOC) = T2MATCNT(I) THEN
                DO; DO J = I2F TO I2L;
                       IF T2COUNT(I) >= RQCOUNT(J) THEN
                         DO; KRQ# = RQ#(J);            /* GENERATE A KEY.  */
                              KCAT = RQCAT(J);
                              KWORD      = VWCRD(IVOC);
                              KMATCNT      = VMATCNT(IVCC);
                              KSECT      = VSECT(IVOC);
                              KDIR@      = VDIR@(IVOC);
                              KVCCUNT      = VCOUNT(IVCC);
                              KVFLAG      = VFLAG(IVOC);
                              KFLAG      = ' ';
                              KMODE      = RQMODE(J);
                              KTYPE = RQTYPE(J);
                              KCOUNT = RQCOUNT(J);
                              KDEPTH = RQDEPTH(J);
                              KVOC@      = (VINCOREKEY*VBLKSIZE) + IVOC;
                              WRITE FILE(CURKEYS)FROM(KEY);
                              KEYCNT = KEYCNT + 1;
                              IF RQKEYL(J) = 'LIST' THEN CALL KEYPRINT;
                        END;
                    END;
                    GO TO VSCAN4X;
               END;
          END;
VSCAN4X:END;
  /* IF THERE ARE SUBSEQUENT BLOCKS, FETCH THE NEXT AND REITERATE.    */
        IF VINCOREKEY < (VKEY(IVK,JVK)+VEXT(IVK,JVK)) THEN
          DO; VKEYR = VINCOREKEY+1;
               CALL READVOC;
               GO TO VSCAN4;
          END;
  /* FINALLY, CHECK THE OVERFLOW BUCKET.                              */
        IF VFLAG(VBLKSIZE-1) = '5' THEN
          DO; VKEYR = VLASTBLK;
               CALL READVOC;
               GO TO VSCAN4;
          END;
PAIRBRKEND: IT2T = 0;                     /* ERASE T2TBL.             */
        T2TBL = -1;
        END TRIPLBRK;                     /* RETURN.                  */

/**********************************************************************/
/**********************************************************************/
```

```
733            ENDTHESAUR:
                       PUT EDIT('***** THESAUR NORMAL TERMINATION',
                                KEYCNT,' SEARCH KEYS GENERATED')
                               (SKIP(2),A,SKIP,F(10),A);
734                    IF MSGCNT = 0
735                      THEN PRT# = '          NO';
736                      ELSE DO; PRT# = MSGCNT; PRT# = SUBSTR(PRT#,5,10); END;
740                    PUT EDIT(PRT#,' WARNING MESSAGES GENERATED ')(SKIP,A,A);
741                    END THESAUR;
```

```
 1            KEYUP:   PROC OPTIONS(MAIN);
              /*********************************************************************/
              /* THIS PROCEDURE PRODUCES THE UNION OF THE KEYS DEVELOPED FROM THE   */
              /* CURRENT SECTION OF TEXT AND THE KEYS FROM EARLIER SECTIONS. THIS   */
              /* BECOMES THE NEW KEYS DATA SET WHICH IS PASSED TO SPRINT.  WE ALSO */
              /* ELIMINATE DUPLICATES IN THE UNION.                                 */
              /* DURING THE MERGE, KFLAG IS SET AS FOLLOWS:                         */
              /*       ' ' - KEYWORD IN BOTH OLD AND CURRENT KEYS                   */
              /*       '*' - KEYWORD IN OLD BUT NOT IN CURRENT KEYS                 */
              /*       '.' - KEYWORD IN CURRENT BUT NOT IN OLD KEYS                 */
              /*********************************************************************/
 2                    DCL OLDKEYS FILE RECORD INPUT;
 3                    DCL CURKEYS FILE RECORD INPUT;
 4                    DCL NEWKEYS FILE RECORD OUTPUT;
 5                    DCL    01 OLDKEY,                 /* OLD KEYS READ AREA.          */
                                 03 OKA      FIXED BIN,
                                 03 OKB      FIXED BIN,
                                 03 OKC      FIXED BIN,
                                 03 OKD      CHAR(1),
                                 03 OKE      CHAR(1),
                                 03 OKF      CHAR(1),
                                 03 OKG      CHAR(1),
                                 03 OKH      FIXED BIN,
                                 03 OKI      FIXED BIN,
                                 03 OKJ      FIXED BIN,
                                 03 OKS1     FIXED BIN,
                                 03 OKS2     CHAR(8),
                                 03 OKS3     FIXED BIN,
                                 03 OKS4     CHAR(18);
 6                    DCL    01 CURKEY,
                                 03 CKA      FIXED BIN,
                                 03 CKB      FIXED BIN,
                                 03 CKC      FIXED BIN,
                                 03 CKD      CHAR(1),
                                 03 CKE      CHAR(1),
                                 03 CKF      CHAR(1),
                                 03 CKG      CHAR(1),
                                 03 CKH      FIXED BIN,
                                 03 CKI      FIXED BIN,
                                 03 CKJ      FIXED BIN,
                                 03 CKS1     FIXED BIN,
                                 03 CKS2     CHAR(8),
                                 03 CKS3     FIXED BIN,
                                 03 CKS4     CHAR(18);
 7                    DCL    01 NEWKEYSORTFLD,               /* SORT FLD OF KEY JUST  */
                               03 NKS1    FIXED BIN,                       /* WRITTEN*/
                                 03 NKS2    CHAR(8),
                                 03 NKS3    FIXED BIN,
                                 03 NKS4    CHAR(18);
 8                    ON ENDFILE (OLDKEYS) BEGIN; OKS1 =  111111111111111B;
11                                                OKS2 = (3)'9';
12                                                OKS3 =  111111111111111B;
13                                                OKS4 = (18)'9';
```

```
                     END;
        ON ENDFILE (CURKEYS) BEGIN; CKS1 =  111111111111111B;
                                    CKS2 = (8)'9';
                                    CKS3 =  111111111111111B;
                                    CKS4 = (18)'9';
                             END;
KUSTART:READ FILE(CURKEYS)INTC(CURKEY);
        IF CKS1 ¬= -1000 THEN /* TEST FOR SPECIAL TEXTSECT RECORD     */
                              /* AND ABORT IF NOT PRESENT.            */
          DO; PUT EDIT('!! FIRST CURKEY IS NOT TEXTSECT RECORD. ',
                       'RUN ABORTED.') (SKIP(2),A,A);
              I=1/0;   /* CAUSE ZERO DIVIDE TO ABCFT                  */
          END;
        IF CKA = 1
          THEN DO; OKS1 = 111111111111111B;     /* FOR 1ST TEXT SECT */
                   OKS2 = (8)'9';               /* THERE ARE NC      */
                   OKS3 = 111111111111111B;     /* CLDKEYS, SO SIMU- */
                   OKS4 = (18)'9';              /* IATE EOF.         */
               END;
          ELSE READ FILE (CLDKEYS) INTO (OLDKEY);
        NKS1, NKS3 = 0;
        NKS2, NKS4 = ' ';
UKCMP:  IF CKS1 < OKS1 THEN GC TO CURLO;
        IF OKS1 < CKS1 THEN GC TO OLDLO;
        IF CKS2 < OKS2 THEN GC TO CURLO;
        IF OKS2 < CKS2 THEN GC TO OLDLO;
        IF CKS3 < OKS3 THEN GC TO CURLO;
        IF OKS3 < CKS3 THEN GO TO OLDLO;
        IF CKS4 < OKS4 THEN GC TC CURLO;
        IF OKS4 < CKS4 THEN GC TO OLDLO;
CURLO:  IF CKS1 = 111111111111111B                  /* MERGE COMPLETE*/
          THEN GC TO ENDKEYUP;
        IF (CKS1=NKS1)&(CKS2=NKS2)&(CKS3=NKS3)&(CKS4=NKS4) /*ELIMINATE*/
          THEN GC TO READCUR;                                /* DUPS.   */
        IF (CKS1¬=OKS1)|(CKS2¬=OKS2)|(CKS3¬=OKS3)|(CKS4¬=OKS4) /*FLAG*/
           THEN CKE = '.';                                   /* NEW KEY */
        NKS1 = CKS1;
        NKS2 = CKS2;
        NKS3 = CKS3;
        NKS4 = CKS4;
        WRITE FILE (NEWKEYS) FROM (CURKEY);
READCUR:READ FILE (CURKEYS) INTO (CURKEY);
        GO TO UKCMP;
CLDLC:  IF (OKS1=NKS1)&(OKS2=NKS2)&(OKS3=NKS3)&(OKS4=NKS4) /*ELIMINATE*/
          THEN GO TO READOLD;                                /* DUPS.   */
        OKE = '*';                          /* FLAG CLD BUT NOT CURRENT */
        NKS1 = OKS1;
        NKS2 = OKS2;
        NKS3 = OKS3;
        NKS4 = OKS4;
        WRITE FILE (NEWKEYS) FROM (OLDKEY);
READOLD:READ FILE (OLDKEYS) INTO (OLDKEY);
        GO TO UKCMP;
```

```
     KEYUE:  PROC OPTIONS(MAIN);

77             ENDKEYUE: END KEYUE;
```

```
STMT LEVEL NEST
  1                SPRINT:  /* VERSION II */ PROCEDURE OPTIONS(MAIN);
                   /*********************************************************************/
                   /********  THIS VERSION IS FOR MEDIUM SIZED THESAURI --        ********/
                   /********  I.E., THOSE WITH A VOCAB TOO LARGE TO FIT IN CORE   ********/
                   /********  BUT WITH A DRCTRY AND THES ONE BLOCK LONG EACH.     ********/
                   /********  IT MAY ALSO BE USED ONLY SLIGHTLY LESS EFFICIENTLY ********/
                   /********  FOR THESAURI WITH DRCTRYS AND THESS THAT HAVE      ********/
                   /********  MANY BLOCKS.  HOWEVER, ONLY ONE BLOCK OF EACH      ********/
                   /********  DATA SET WILL BE IN CORE AT A TIME.                ********/
                   /*********************************************************************/
                   /* THIS IS THE SEARCH-AND-PRINT PHASE OF VIA.                        */
                   /* WE PROCESS THE KEYS AS FOLLOWS:                                   */
                   /*         DO A THESAURUS SEARCH DOWN THRU SEVERAL LEVELS, FOR EACH  */
                   /*         KEY.                                                      */
                   /*         THERE ARE FIVE SEARCH MODES:                              */
                   /*              A: TEXT-LIMITED - ALL NODES IN TEXT.                 */
                   /*              B: TEXT-ORIENTED - ROOT AND LEAVES IN TEXT.          */
                   /*                             INTERMEDIATE NODES MAY OR MAY NOT BE. */
                   /*              C: TEXT-ROOTED - ROOT IN THE TEXT.                   */
                   /*              D: TEXT-RELATED - LEAVES IN THE TEXT.                */
                   /*              E: THESAURUS - NOTHING IS REQUIRED TO BE IN THE TEXT.*/
                   /*                                                                   */
                   /* THE THESAURUS IS A RING STRUCTURE.  WE PRINT IT, HOWEVER, AS A    */
                   /* REDUNDANT TREE.  SOME REDUNDANCY IS OMITTED BY THE FOLLOWING      */
                   /* RESTRICTION: NO NODE (WORD OR CATEGORY) APPEARS MORE THAN ONCE IN */
                   /* ANY PATH.                                                         */
                   /*                                                                   */
                   /* THE DEPTH OF SEARCH IN THE THESAURUS IS CONTROLLED BY THE "DEPTH" */
                   /* PARAMETER ORIGINALLY ENTERED IN THE ANALYSIS REQUEST CARD TO THE  */
                   /* "THESAUR" PROGRAM, AND PASSED TO THIS PROGRAM IN THE KEY RECORD.  */
                   /* DEPTH IS USED TO LIMIT THE LEVEL OF RECURSION OF THE SUBROUTINE   */
                   /* "WORD".  THUS, DEPTH IS THE NUMBER OF LEVELS IN THE PRINTED TREE. */
                   /* THE DEFAULT VALUE OF DEPTH IS 3.  MAXIMUM ALLOWABLE IS 9.         */
                   /*                                                                   */
                   /* PRINTING CONVENTIONS:                                             */
                   /*      1. WORDS NOT IN TEXT ARE ALWAYS ENCLOSED IN ().              */
                   /*      2. WORDS APPEARING FOR THE FIRST TIME ARE PRECEDED BY A DASH */
                   /*         LINE: --------.                                           */
                   /*      3. KEYS APPEARING FOR THE FIRST TIME AS A SEARCH KEY ARE     */
                   /*         PRECEDED BY A PERIOD: ..                                  */
                   /*      4. KEYS WHICH DO NOT QUALIFY AS SEARCH KEYS IN THE CURRENT   */
                   /*         SECTION OF TEXT, BUT WHICH HAVE BEEN KEYS IN EARLIER      */
                   /*         SECTIONS, ARE PRECEDED BY AN ASTERISK: *.                 */
                   /*      5. FOR BREVITY, FOR EACH TREE PRINTED, IF A CATEGORY APPEARS */
                   /*         MORE THAN ONCE AT THE DEEPEST LEVEL OF THE TREE, THE LIST */
                   /*         OF WORDS IN THE CAT IS ELIDED FOR ALL APPEARANCES OF THE  */
                   /*         CAT SUBSEQUENT TO THE FIRST.  (SEE "CATX".)               */
                   /*      5. AT THE COMPLETION OF EACH TREE, A LIST OF ALL CATEGORIES  */
                   /*         APPEARING IN THE TREE IS PRINTED.  (SEE "CATS").          */
                   /*      7. ON EACH LEVEL, TEMP ENTRIES ARE LISTED ONLY THE FIRST TIME */
                   /*         THEIR MATCNT APPEARS.  (SEE "MCNTBL".)                    */
                   /*                                                                   */
                   /*********************************************************************/
  2      1                DCL VOCAB  FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,
                              DRCTRY FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,
                              THES   FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,
STMT LEVEL NEST

                           THSCTL FILE STREAM INPUT,
                           KEYS   FILE RECORD INPUT;
  3     1              DCL   VBLKSIZE  FIXED BIN,
                             DBLKSIZE  FIXED BIN,
                             TBLKSIZE  FIXED BIN,
                             VLASTBLK  FIXED BIN,
                             DLASTBLK  FIXED BIN,
                             TLASTBLK  FIXED BIN,
                             VKEY      FIXED DEC(5),
                             DKEY      FIXED DEC(5),
                             TKEY      FIXED DEC(5),
                             VINCOREKEY  FIXED DEC (5) INITIAL(-1),
                             DINCOREKEY  FIXED DEC (5) INITIAL(-1),
                             TINCOREKEY  FIXED DEC (5) INITIAL(-1);
  4     1              DCL   VKEYS(26,26)    FIXED DEC(3),  /* VOCAB BUCKET KEYS      */
                             VEXTS(26,26)    FIXED DEC(3);  /* VOCAB BUCKET EXTENTS   */
  5     1              DCL   01 VOCBLOCK CONTROLLED,  /* FOR READING IN VOCAB 1ST    */
                                02 VOC   (0:VBLKSIZE-1),
                                   03  VMATCNT    FIXED BIN,
                                   03  VSECT      FIXED BIN,
                                   03  VDIR@      FIXED BIN,
                                   03  VCOUNT     FIXED BIN,
                                   03  VX         CHAR(1),
                                   03  VFLAG      CHAR(1),
                                   03  VWORD      CHAR(18);
  6     1              DCL   01 DIRBLOCK CONTROLLED,
                                02 DIR   (0:DBLKSIZE-1),
                                   03   DCAT      CHAR(8),
                                   03   DTHS@     FIXED BIN,
                                   03   DLNG      FIXED BIN;
  7     1              DCL   01 THSBLOCK CONTROLLED,
                                02 THS  (0:TBLKSIZE-1),
                                   03   TDIR@     FIXED BIN,
                                   03   TVOC@     FIXED BIN;
  8     1              DCL   01 KEY,                   /* SEARCH AND PRINT KEY         */
                                   02   KSECT     FIXED BIN,
                                   02   KDIR@     FIXED BIN,
                                   02   KVCOUNT   FIXED BIN,
                                   02   KVFLAG    CHAR (1),
                                   02   KFLAG     CHAR(1),
                                   02   KMODE     CHAR(1),
                                   02   KTYPE     CHAR (1),
                                   02   KCOUNT    FIXED BIN,
                                   02   KVOC@     FIXED BIN,
                                   02   KDEPTH    FIXED BIN,
                                   02   KRQ#      FIXED BIN,
                                   02   KCAT      CHAR(8),
                                   02   KMATCNT   FIXED BIN,
                                   02   KWORD     CHAR (18);
  9     1              DCL   PATH(11)  FIXED BIN;  /* THE PATH OF NODES TO BE         */
                                                  /* PRINTED.  INDEXED BY LPATH.     */
 10     1              DCL   WORDSP(11)  CHAR(21) VARYING;  /* WORDS SAVED FOR       */
                                                            /* POSSIBLE PRINTING.    */
                                                            /* INDEXED BY LPATH.     */
```

```
        SPRINT:  /* VERSION II */ PROCEDURE OPTIONS(MAIN);                                                  PAGE      4

 STMT LEVEL NEST

  11      1               DCL   CATSP(10)    CHAR(8);             /* CATEGORIES SAVED FOR */
                                                                  /* POSSIBLE PRINTING.   */
                                                                  /* INDEXED BY LPATH.    */
  12      1               DCL   COUNTSP(11)  FIXED BIN;           /* FREQ COUNTS FOR      */
                                                                  /* WORDSP.              */
                                                                  /* INDEXED BY LPATH.    */
  13      1               DCL   (CURCAT,        /* INDEX IN DIR OF 1ST CAT                */
                                PRINTNDX,       /* LAST POS IN PATH THAT HAS OR IS TO     */
                                                /* BE PRINTED.  IT IS ALWAYS EVEN, FOR A*/
                                                /* CAT IS PRINTED WHENEVER THE PRECEDING*/
                                                /* WORD HAS BEEN.                         */
                                LEVEL)          /* LEVEL OF RECURSION                     */
                                           FIXED BIN;
  14      1               DCL (P1, P2) CHAR(1);           /* PARENS FOR WORDS NOT IN TEXT*/
  15      1               DCL   CURWDX     FIXED BIN;      /* FOR PRINTING CURWD        .*/
  16      1               DCL   DEPTH      FIXED BIN INITIAL(-1);    /* RECURSION DEPTH. */
                                                           /* SEE COMMENTS FOR ALTERING. */
  17      1               DCL   N          FIXED BIN;      /* PRINT TAB.                 */
  18      1               DCL   DUMM  FIXED BIN INITIAL(-1);  /* DUMMY PARM FOR CALLING*/
                                                              /* TEMPRNT FROM MAIN LOOP*/
  19      1               DCL   TEXTSECT  FIXED BIN;          /* CURRENT SECT OF TEXT. */
  20      1               DCL   DASH      CHAR(8) VARYING;    /* DASH LINE FOR PRINTING*/
  21      1               DCL   DASHC     CHAR(8) INITIAL('--------'); /* FOR RESETTING*/
                                                                        /* DASH.       */
  22      1               DCL   CH1       CHAR(1);            /* SINGLE CHARACTER      */
  23      1               DCL   CATX(1000) FIXED BIN;      /* SEE PRINTING CONV #5.      */
  24      1               DCL   (ICATX,ICATXMAX)FIXED BIN INITIAL(0); /* CATX INDICES  */
  25      1               DCL   CATS(1000) FIXED BIN;      /* SEE PRINTING CONV #5.      */
  26      1               DCL   (ICATS,ICATSMAX)FIXED BIN INITIAL(0); /* CATS INDICES  */
  27      1               DCL   PGNO   FIXED BIN;          /* PAGE NUMBER                */
  28      1               DCL   KRQ#SAV  FIXED BIN INITIAL(-1); /*FOR BREAK ON KRQ#    */

                   /*******************************************************************/
                   /*******************************************************************/

  29      1        GETCTL: GET FILE(THSCTL) DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE,
                                             VLASTBLK,DLASTBLK,TLASTBLK);
  30      1               ALLOCATE VOCBLOCK, DIRBLOCK, THSBLOCK;
  31      1               GET FILE(THSCTL) EDIT (VKEYS,VEXTS)(F(4));
  32      1               READ FILE(KEYS)INTO(KEY);        /* READ 1ST KEY - A DUMMY     */
  33      1               TEXTSECT = KSECT;                /* RECORD CONTAINING CURRENT  */
                                                           /* TEXTSECT NUMBER.           */
  34      1               ON ENDPAGE(SYSPRINT) CALL PGHDG;

                   /*******************************************************************/
                   /*******************************************************************/

                   /* MAJOR LOOP -                                                    */
                   /* PROCESS THE KEYS FILE SEQUENTIALLY.                             */
  36               ON ENDFILE(KEYS)GO TO SPRINTEND;
  38               READKEY: READ FILE (KEYS) INTO (KEY);
                      /* WE BYPASS THIS WORD IF THE SEARCH MODE IS A,B, OR C AND THE  */
                      /* WORD IS NEITHER IN THE TEXT NOR AMONG THE PREVIOUS KEYS.     */
  39      1                   IF ((KMODE='A    (KMODE='B')|(KMODE='C'))
        SPRINT:  /* VERSION II */ PROCEDURE OPTIONS(MAIN);                                                PAGE     5

 STMT LEVEL NEST

                                 & (KFLAG¬='*')
                                 & ((KMATCNT=-1)|(KVFLAG='2'))
  40      1                       THEN GO TO READKEY;
                   /* HERE WE KNOW THE WORD IS A VALID ROOT.                                         */
  41      1              IF KRQ# ¬= KRQ#SAV
  42      1                THEN DO; PGNO = 1;               /* BRAND NEW ANALYSIS!                   */
  44      1                           KRQ#SAV = KRQ#;
  45      1                      END;
  46      1                ELSE PGNO = PGNO + 1;            /* NEW TREE; SAME ANALYSIS.              */
  47      1              PUT EDIT((34)'*') (PAGE,A);                                  /* NEW PAGE */
  48      1              PUT EDIT('ANALYSIS ',KRQ#,'PAGE',PGNO)
                                  (SKIP,A,F(5),X(97),A,F(4));
  49      1              PUT EDIT('MODE ',KMODE,',  TYPE ',KTYPE,'.  SEARCH KEY IS ')
                                  (SKIP,COLUMN(20),5 A);
  50      1              IF ((KTYPE='1')|(KTYPE='3'))
  51      1                THEN PUT EDIT(KCAT) (A(8));
  52      1                ELSE IF KTYPE='2'
  53      1                       THEN PUT EDIT('ROOT OF "',KWORD,'"')
                                             (A,A(LENGTH(KWORD)),A);
  54      1                       ELSE IF KTYPE='4'
  55      1                              THEN PUT EDIT('"',KWORD,'"')
                                                    (A,A(LENGTH(KWORD)),A);
  56      1              IF KTYPE = '1'
  57      1                THEN PUT EDIT(' AND OCCURS',KVCOUNT,' TIMES')
                                        (A,F(4),A);
  58      1              IF KTYPE = '2'
  59      1                THEN PUT EDIT(' AND OCCURS AT LEAST',KCOUNT,' TIMES')
                                        (A,F(4),A);
                  /* ESTABLISH SEARCH DEPTH.                                                          */
  60      1       SET_DEPTH:  DEPTH = KDEPTH;
  61      1              IF DEPTH = -1
  62      1                THEN DO; DEPTH = 3;           /**** DEFAULT SEARCH DEPTH ****/
  64      1                          PUT EDIT('SEARCH DEPTH = 3, BY DEFAULT')
                                              (SKIP,COLUMN(20),A);
  65      1                          GO TO SETN;
  66      1                     END;
  67      1              IF DEPTH > 9 THEN
  68      1                DO; DEPTH = 9;
  70      1                      PUT EDIT('MAXIMUM PERMISSIBLE SEARCH DEPTH IS 9.')
                                          (SKIP,COLUMN(20),A);
  71      1                END;
  72      1              PUT EDIT('SEARCH DEPTH =',DEPTH) (SKIP,COLUMN(20),A,F(2));
  73      1       SETN:  IF DEPTH > 5 THEN N=4; ELSE N=8;     /* SET PRINT TAB.                  */
  76      1              DASH = SUBSTR(DASHC,1,N);  /* SET DASH TO PROPER LENGTH.               */
  77      1              CALL NOPHDG;                        /* PRINT STANDARD PAGE HEADING*/
  78      1              PUT SKIP;         /* NOW TO DO SOMETHING WITH THE ROOT:                */
  79      1              IF (KMATCNT=-1)|(KVFLAG='2')
  80      1                         THEN DO; P1='('; P2=')'; END;
  84      1                         ELSE DO; P1=' '; P2=' '; END;
  88      1              IF (KMODE='D')&(KFLAG¬='*')&(KMATCNT=-1)
                                  THEN /* SAVE FOR POSSIBLE PRINTING LATER.                          */
  89      1                            DO; WORDSP(1) = KFLAG     ||P1||KWORD     ||P2;
  91      1                                 COUNTSP(1) = KVCOUNT;
  92      1                                 PRINTNDX = 0;
```

```
STMT LEVEL NEST

  93     1                               END;
  94     1                          ELSE /* PRINT IT NOW.                                  */
  94     1                               DO; PUT EDIT(KFLAG,P1,KWORD,P2)
                                                      (SKIP,COLUMN(N),4 A);
  96     1                                   IF KVCOUNT ¬= -1
  97     1                                     THEN PUT EDIT(KVCOUNT)(F(5));
  98     1                                   PRINTNDX = 2;  /* SIGNAL THE PRINTING    */
                                                            /* OF THE CATEGORIES ON   */
                                                            /* THE NEXT LEVEL.        */
  99     1                               END;
 100     1                ICATX,ICATXMAX = 0;            /* RESET CATX.               */
 101     1                PATH = 0;                      /* RESET PATH VECTOR         */
 102     1                PATH(1) = KVOC@;
 103     1                LPATH = 1;
 104     1                LEVEL = 1;
 105     1                IF KVFLAG > '2' THEN GO TO TEMPKEY; /* SEE IF THIS IS A TEMP */
                                                            /* VOCAB ENTRY(VFLAG='3')*/
 107     1                IF ((KVFLAG='1')|(KVFLAG='2'))       /* FIND & PRINT ANY     */
 108     1                  THEN CALL TEMPRNT(VINCOREKEY,DUMM);  /* TEMP VOCAB ENTRIES */
                                          /* DIRECTLY BENEATH THE SEARCH KEYWORD.  */
 109     1                IPARM1 = KVOC@;
 110     1                IPARM2 = KDIR@;
 111     1                IPARM3 = IPARM2;
 112     1                CALL WORD(IPARM1,IPARM2,IPARM3);
 113     1                GO TO NEXTKEY;
 114     1          TEMPKEY:
                      /* HERE THE KEY HAPPENS TO BE A TEMP VOCAB ENTRY.                 */
                          IVK = UNSPEC(SUBSTR(KWORD,1,1));  /* IDENTIFY BUCKET.         */
 115     1                JVK = UNSPEC(SUBSTR(KWORD,2,1));
 116     1                IF IVK < 202 THEN IVK = IVK - 192;
 118     1                  ELSE IF IVK < 218 THEN IVK = IVK - 199;
 120     1                    ELSE IVK = IVK - 207;
 121     1                IF JVK < 202 THEN JVK = JVK - 192;
 123     1                  ELSE IF JVK < 218 THEN JVK = JVK - 199;
 125     1                    ELSE JVK = JVK - 207;
                      /* FIRST, SCAN THE END OF THE BUCKET & THE OVFL BUCKET FOR OTHER  */
                      /* TEMP ENTRIES WITH SAME MATCNT AND PRINT THEM DIRECTLY BENEATH  */
                      /* THE SEARCH KEYWORD.                                            */
                      /* FETCH THE LAST BLOCK IN THE BUCKET.                            */
 126     1                VKEY = VKEYS(IVK,JVK) + VEXTS(IVK,JVK);
 127     1                IF VKEY ¬= VINCOREKEY THEN
 128     1                  DO; READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
 130     1                      VINCOREKEY = VKEY;
 131     1                  END;
 132     1          VSCAN1: DO IVOC = 0 TO VBLKSIZE-1 WHILE(VWORD(IVOC)¬=' ');
 133     1    1             IF ((VFLAG(IVOC)>'2') & (VMATCNT(IVOC)=KMATCNT)
                             & (VWORD(IVOC)¬=KWORD)) THEN
 134     1    1               DO; IF PRINTNDX = 0 THEN
 136     1    1                     DO; PRINTNDX = 2;    /* PRINT THE KEY WE HAVE BEEN */
 138     1    1                         PUT EDIT(WORDSP(1))              /* SAVING.      */
                                                 (SKIP,COLUMN(N),A);
 139     1    1                         IF COUNTSP(1) ¬= -1
 140     1    1                           THEN PUT EDIT(COUNTSP(1))(F(5));
 141     1    1                     END;
 STMT LEVEL NEST

 142     1     1                  IF VSECT(IVOC)=TEXTSECT THEN CH1='-'; ELSE CH1=' ';
 145     1     1                  PUT EDIT(CH1,' ',VWORD(IVOC),' ')    /* PRINT THE TEMP*/
                                          (SKIP,COLUMN(N),4 A);
 146     1     1                  IF VCOUNT(IVOC) ¬= -1
 147     1     1                    THEN PUT EDIT(VCOUNT(IVOC))(F(5));
 148     1     1              END;
 149     1     1          END;
                    /* CHECK OVERFLOW BUCKET.                                          */
 150     1                IF VFLAG(VBLKSIZE-1) = '5' THEN
 151     1                  DO; VKEY = VLASTBLK;
 153     1                      READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
 154     1                      VINCOREKEY = VKEY;
 155     1                      GO TO VSCAN1;
 156     1                  END;
                    /* FINALLY, SCAN THE BUCKET AND USE EACH PERMANENT WORD WITH THE   */
                    /* SAME MATCNT TO CALL "WORD".                                     */
 157     1                VKEY = VKEYS(IVK,JVK);
 158     1                IF VKEY ¬= VINCOREKEY THEN
 159     1                  DO; READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
 161     1                      VINCOREKEY = VKEY;
 162     1                  END;
 163     1        VSCAN2: DO IVOC = 0 TO VBLKSIZE-1       /* SCAN BUCKET SEQUENTIALLY.  */
                              WHILE ((VWORD(IVOC) ¬= ' ') & (VFLAG(IVOC) < '3'));
 164     1     1            IF VMATCNT(IVOC) = KMATCNT THEN   /* IF MATCNTS MATCH, KEE */
 165     1     1              DO;                             /* A SEARCH ON THIS WORD */
                                /* THE FOLLOWING LOGIC TO DETERMINE PRINTING IS THE  */
                                /* SAME AS THAT IN THE SECTION, "CATEGORY", SINCE    */
                                /* THIS WORD IS NOT THE KEY.                         */
 166     1     1                PATH(1) = (VKEY*VBLKSIZE) + IVOC;
 167     1     1                IF VSECT(IVOC)=TEXTSECT THEN CH1='-'; ELSE CH1=' ';
 170     1     1                IF VFLAG(IVOC) = '2'
 171     1     1                  THEN DO; P1='('; P2=')'; END;
 175     1     1                  ELSE DO; P1=' '; P2=' '; END;
 179     1     1                IF ((KMODE='B')|(KMODE='C'))&(VFLAG(IVOC)='2')
                                  THEN /* SAVE FOR POSSIBLE PRINTING AT A LATER LEVEL*/
 180     1     1                    DO; WORDSP(LPATH) = CH1||P1||VWORD(IVOC)||P2;
 182     1     1                        COUNTSP(LPATH) = VCOUNT(IVOC);
 183     1     1                    END;
 184     1     1                  ELSE /* PRINT IT NOW                               */
 184     1     1                    DO; IF PRINTNDX = 0 THEN      /* PRINT SAVED KEY  */
 186     1     1                             IF COUNTSP(1) ¬= -1 THEN
 187     1     1                               PUT EDIT(COUNTSP(1))(F(5));
 188     1     1                          DO; PUT EDIT(WORDSP(1))(SKIP,COLUMN(N),A);
 190     1     1                          END;
 191     1     1                        PUT EDIT(CH1,P1,VWORD(IVOC),P2)
                                               (SKIP,COLUMN(N),4 A);
 192     1     1                        IF VCOUNT(IVOC) ¬= -1 THEN
 193     1     1                          PUT EDIT(VCOUNT(IVOC))(F(5));
 194     1     1                        PRINTNDX = 2;
 195     1     1                    END;
 196     1     1                IPARM1 = (VKEY*VBLKSIZE) + IVOC;
 197     1     1                IPARM2 = VDIR@(IVOC);
 198     1     1                IPARM3 = IPARM2;
 199     1     1                CALL WORD(IPARM1,IPARM2,IPARM3);
```

```
STMT LEVEL NEST

200     1    1                    IF VKEY ¬= VINCOREKEY THEN
201     1    1                      DO: READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
203     1    1                          VINCOREKEY = VKEY;
204     1    1                      END;
205     1    1                  END;
206     1    1              END;
                         /* CONTINUE WITH SUBSEQUENT BLOCKS IN BUCKET, IF ANY.              */
207     1                   IF VKEY < VKEYS(IVK,JVK)+VEXTS(IVK,JVK) THEN
208     1                     DO: VKEY = VKEY + 1;
210     1                         GO TO VSCAN2;
211     1                     END;
212     1           NEXTKEY:
                            CALL CATSYP;                    /* PRINT THE CATEGORY SYNOPSIS*/
213     1                   GO TO READKEY;

                /*******************************************************************/
                /*************                                ***********************/
                /************* S U B R O U T I N E S ***********************/
                /*************                                ***********************/
                /*******************************************************************/

214     1           WORD:   PROC(CURWORD,CHAINHD,DIR@) RECURSIVE;
                    /* A RING OF CATEGORIES */
215     2                   DCL  (CURWORD,  /* NDX IN VOC OF CURRENT WORD                  */
                                  CHAINHD,  /* NDX IN DIRECTORY OF CURRENT CATEGORY        */
                                            /* ALSO HEAD OF CHAIN WHICH IS WORD RING       */
                                  DIR@,     /* NDX IN DIRECTORY OF NEXT CATEGORY           */
                                  THS@,     /* NDX IN THESAURUS OF CURRENT WORD            */
                                  VOC@,     /* CURWORD FOR NEXT LEVEL OF RECURSION, TEMP? */
                                  DIR@1,    /* CHAINHD FOR NEXT LEVEL OF RECURSION         */
                                  DIR@2,    /* ID FOR NEXT LEVEL OF RECURSION              */
                                  NEXTCAT)  /* NEXTCAT IN CURRENT LEVEL OF RECURSION       */
                                      FIXED BIN;
216     2                   DCL  VKEY  FIXED DEC(5);  /* VOCAB KEY; MUST BE LOCAL.         */
217     2                   DCL  DKEY  FIXED DEC(5);  /* DRCTY KEY; MUST BE LOCAL.         */
218     2                   DCL  TKEY  FIXED DEC(5);  /* THES  KEY; MUST BE LOCAL.         */
219     2                   DCL (IVOC,      /* VOCBLOCK INDEX - THESE INDICES MUST BE      */
                                 IDIR,      /* DIRBLOCK INDEX            LOCAL TO THE      */
                                 ITHS)      /* THSBLOCK INDEX            RECURSION LEVEL */
                                      FIXED BIN;
220     2                   DCL J FIXED BIN;  /* LOOP CONTROL LOCAL TO RECURSION LEVEL     */
221     2                   DCL  MCTBL(100) FIXED BIN;      /* RECORD OF MATCHES THAT HAVE*/
                                                            /* BEEN SEARCHED FOR TEMPORARY*/
                                                            /* VOCAB ENTRIES ON THIS LEVEL*/
                                                            /* SEE PRINTING CONV. #7.     */
222     2                   DCL  IMCMAX  FIXED BIN;         /* MCTBL SIZE.                */
223     2                   IMCMAX = 0;
                      /* WE RING AROUND THE CATEGORIES */
224     2                   LPATH = LPATH + 1;
225     2                   IF CURWORD = -1                 /* IF THIS IS A CAT KEY THERE */
226     2                     THEN NEXTCAT = CHAINHD;       /* WILL BE NC CURWORD AND ONLY*/
                                                            /* ONE CAT - THE ROOT.  WE    */
                                                            /* MUST SAVE ITS @ IN NEXTCAT.*/
227     2                   IF (LEVEL=1) | ((LEVEL=2)&(KVFLAG>'2')) THEN GO TO DIRFETCH;
```

```
STMT LEVEL NEST

229      2          WORDRING: IF DIR@ = CHAINHD THEN GO TO WORDEND:  /* SEE IF WE HAVE    */
                                                                     /*   COMPLETED THE  */
                                                                     /*   RING.          */
231      2          DIRFETCH:
                    /* AT THIS POINT WE KNOW WHICH DRCTRY & THES BLOCKS WE WILL NEED IN  */
                    /* CORE, SO WE GET THEM.  FIRST WE LOOK TO SEE IF THEY ARE ALREADY IN*/
                    /* CORE, AND WE FETCH ONLY IF THEY ARE NOT.                          */
                    /* FOR VERY LARGE DATA SETS, A VERY EFFICIENT METHOD WOULD BE TO     */
                    /* ALLOCATE IN CORE, FOR EACH DATA SET, AS MANY BLOCKS AS THERE ARE  */
                    /* LEVELS OF RECURSION.                                              */
                            DKEY = DIR@ / DBLKSIZE;
232      2                  IDIR = DIR@ - (DKEY*DBLKSIZE);
233      2                  IF DKEY ¬= DINCOREKEY THEN
234      2                    DO; READ FILE(DRCTRY) INTO(DIRBLOCK) KEY (DKEY);
236      2                        DINCOREKEY = DKEY;
237      2                    END;
238      2          THSFETCH:
                    /* THE REMARKS UNDER DIRFETCH ALSO APPLY HERE.                       */
                            THS@ = DTHS@(IDIR);  /* PICK UP THES INDEX FROM THE DRCTRY.  */
239      2                  TKEY = THS@ / TBLKSIZE;
240      2                  ITHS = THS@ - (TKEY*TBLKSIZE);
241      2                  IF TKEY ¬= TINCOREKEY THEN
242      2                    DO; READ FILE(THES) INTO(THSBLOCK) KEY (TKEY);
244      2                        TINCOREKEY = TKEY;
245      2                    END;
246      2          LOOKBACK: DO I = 2 TO (LPATH-1) BY 2;  /* SEE IF THIS CAT IS ALREAD( */
247      2     1                  IF DIR@ = PATH(I) THEN   /* IN PATH. IF SO, BYPASS IT: */
248      2     1                      GO TO CATSKIP;
249      2     1              END;
250      2                    GO TO GOODCAT;
251      2          CATSKIP:                ITHS = ITHS - 1;          /* FIND NEXT CAT     */
252      2                                  DO J = 1 TO DLNG(IDIR);   /*   AND GO ON.      */
253      2     1                              ITHS = ITHS + 1;
254      2     1                              IF ITHS = TBLKSIZE THEN /* IF CAT OVERFLOWS */
255      2     1                                DO; TKEY = TKEY + 1;  /* TO NEXT BLOCK,    */
257      2     1                                    ITHS = 0;         /* MUST FETCH IT.    */
258      2     1                                    READ FILE(THES)INTO(THSBLOCK) KEY(TKEY);
259      2     1                                    TINCOREKEY = TKEY;
260      2     1                                END;
261      2     1                              IF TVOC@(ITHS) = CURWORD THEN
262      2     1                                DO; DIR@ = TDIR@(ITHS);
264      2     1                                    GO TO WORDRING;
265      2     1                                END;
266      2     1                            END;
                                            /* MUST NEVER REACH HERE!!                 */
267      2                                  PUT EDIT('!! WORD ENTRY NOT FOUND IN CAT IN ',
                                                     'THES. RUN ABORTED.') (SKIP (2),A,A);
268      2                                  CURWDX = CURWORD;
269      2                                  PUT DATA(I,J,DKEY,TKEY,DTHS@(IDIR),DLNG(IDIR),
                                                     TVOC@(ITHS),CURWDX,LEVEL);
270      2                                  GO TO SPRINTEND;
271      2          GOODCAT:
                        /* HERE WE KNOW THE CAT IS NOT REDUNDANT, SO PUT IT IN THE TREE..*/
                            PATH(LPATH) = DIR@;
```

```
STMT LEVEL NEST

                      /* THE DISPOSITION OF A CATEGORY IS ALWAYS THE SAME AS THAT OF   */
                      /* THE WORD ON THE NEXT HIGHER LEVEL .  THE VALUE OF PRINTNDX    */
                      /* TELLS US.                                                      */
272     2                 IF PRINTNDX = LPATH
273     2                       THEN PUT EDIT(DCAT(IDIR))(SKIP,COLUMN(N*LPATH),A(3));
274     2                       ELSE CATSP(LPATH) = DCAT(IDIR);
                      /* PRINTING CONV #555.  NOW LOOK AT THE HISTORY OF THE CAT IN    */
                      /* THIS TREE.                                                     */
275     2                 IF LEVEL = DEPTH THEN           /* CHECK CATX ONLY AT DEEPEST */
276     2                   DO; DO ICATX = 1 TO ICATXMAX;            /* LEVEL           */
278     2    1                    IF CATX(ICATX) = DIR@ THEN GO TO CATSKIP;
280     2    1                  END;
281     2                       IF ICATXMAX < 1000 THEN
282     2                         DO; ICATXMAX = ICATXMAX + 1;
284     2                             CATX(ICATXMAX) = DIR@;
285     2                         END;
286     2                   END;
287     2                 DO ICATS = ICATSMAX TO 1 BY -1; /* NOW RECORD ITS HISTORY    */
288     2    1              IF CATS(ICATS) ≐ DIR@ THEN GO TO HISTX;
290     2    1            END;
291     2                 IF ICATSMAX < 1000 THEN          /* NEW CATEGORY.             */
292     2                   DO; ICATSMAX = ICATSMAX + 1;
294     2                       CATS(ICATSMAX) = DIR@;
295     2                   END;
296     2         HISTX:  ;
                  /*********************************************************************/
297     2         CATEGORY:
                  /* A RING OF WORDS */
                          LPATH = LPATH + 1;
298     2         CAT:    ITHS = ITHS - 1;
299     2                 DO J = 1 TO DLNG(IDIR);          /* RING AROUND THE WORDS.    */
300     2    1              ITHS = ITHS + 1;
301     2    1              IF ITHS = TBLKSIZE THEN        /* IF CAT OVERFLOWS TO NEXT  */
302     2    1                DO; TKEY = TKEY + 1; ITHS = 0;        /* BLOCK WE MUST     */
305     2    1                    READ FILE(THES)INTO(THSBLOCK)KEY(TKEY);  /* FETCH IT */
306     2    1                    TINCOREKEY = TKEY;
307     2    1                END;
308     2    1              VOC@ = TVOC@(ITHS);
309     2    1                IF CURWORD = VOC@ THEN NEXTCAT = TDIR@(ITHS);  /* SAVE    */
                                                            /* NEXT CAT WHEN WE SEE IT */
311     2    1              DO I = 1 TO (LPATH-1) BY 2;  /* SEE IF THIS WORD IS ALREADY*/
312     2    2                IF VOC@ = PATH(I)            /* IN PATH & BYPASS IF SO    */
313     2    2                  THEN GO TO CATEND;
314     2    2              END;
315     2    1    VOCFETCH: VKEY = VOC@ / VBLKSIZE;        /* FETCH VOCAB ENTRY         */
316     2    1              IVOC = VOC@ - (VKEY*VBLKSIZE);
317     2    1              IF VKEY ¬= VINCOREKEY THEN
318     2    1                DO; READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
320     2    1                    VINCOREKEY = VKEY;
321     2    1                END;
322     2    1                  IF VFLAG(IVOC) > ' ' THEN      /* CHECK FOR TEMP ENTRIES*/
323     2    1                    DO; DO I = 1 TO IMCMAX;
325     2    2                          IF MCTBL(I) = VMATCNT(IVOC) THEN GO TO PRMVOC;
327     2    2                        END;
```

```
STMT LEVEL NEST

                                      /* HERE THERE ARE TEMP  (VFLAG='3') ENTRIES TO        */
                                      /* PRINT.  SINCE THEY ARE ALWAYS IN THE TEXT,         */
                                      /* THEY MAY BE PRINTED IMMEDIATELY.                    */
 328     2     1                      MCTBL(I) = VMATCNT(IVOC);     /* INSERT NEW MATCNT*/
 329     2     1                      IMCMAX = I;                   /* IN MCTBL.         */
 330     2     1                      CALL TEMPRNT(VKEY,IVOC);      /* PRINT THEM.       */
 331     2     1                   END;
                 /* NOW BACK TO THE PERMANENT VOCAB ENTRY.                                   */
 332     2     1  PRMVOC:       IF KMODE    ='A'          /* BYPASS IF MODE A AND WORD */
                                 &  (VMATCNT(IVOC)=-1)                        /* NOT IN  */
 333     2     1                 THEN GO TO CATEND;                           /* TEXT.   */
                 /* HERE WE KNOW THE WORD IS NOT REDUNDANT, SO PUT IT IN THE TREE.*/
 334     2     1                PATH(LPATH) = VOC@;
                 /* NOW, FOR MODES B & D, WE PRINT ONLY IF THIS IS IN THE TEXT.      */
                 /* OTHERWISE, WE SAVE IT FOR POSSIBLE PRINTING AT A LATER LEVEL. */
 335     2     1                IF KMODE    ='B' | KMODE    ='D'                            .
 336     2     1                   THEN IF  (VMATCNT(IVOC)=-1)
                                         THEN /* SAVE FOR POSSIBLE PRINTING AT A        */
                                              /* LATER LEVEL.                           */
 337     2     1                              DO; WORDSP(LPATH)='  ('||VWORD(IVOC)||')';
 339     2     1                                  COUNTSP(LPATH) = VCOUNT(IVOC);
 340     2     1                                  GO TO NXTLVL;
 341     2     1                              END;
 342     2     1                         ELSE /* PRINT THE WHOLE PATH WE HAVE BEEN      */
                                              /* SAVING.                                */
 342     2     1                              DO; DO I = (PRINTNDX+1) TO (LPATH-1);
 344     2     2                                         PUT EDIT (WORDSP(I))
                                                          (SKIP,COLUMN(N*I),A);
 345     2     2                                         IF COUNTSP(I) ¬= -1 THEN
 346     2     2                                           PUT EDIT(COUNTSP(I)) (F(5));
 347     2     2                                         I = I + 1;
 348     2     2                                         PUT EDIT (CATSP(I))
                                                          (SKIP,COLUMN(N*I),A);
 349     2     2                                  END;
 350     2     1                              END;
 351     2     1                IF  (VMATCNT(IVOC)=-1)|(VFLAG(IVOC)='2')
 352     2     1                     THEN DO; P1='('; P2=')'; END;
 356     2     1                     ELSE DO; P1=' '; P2=' '; END;
 360     2     1                PUT SKIP;
 361     2     1                IF VSECT(IVOC)  = TEXTSECT
 362     2     1                  THEN PUT EDIT  (DASH)(COLUMN(N*(LPATH-1)),A);
 363     2     1                PUT EDIT(' ',P1,VWORD(IVOC),P2)(COLUMN(N*LPATH),4 A);
 364     2     1                IF VCOUNT(IVOC) ¬= -1 THEN
 365     2     1                  PUT EDIT(VCOUNT(IVOC)) (F(5));
 366     2     1                PRINTNDX = LPATH + 1;
 367     2     1       NXTLVL: IF LEVEL < DEPTH THEN
 368     2     1                     DO;  LEVEL = LEVEL + 1;
 370     2     1                          DIR@2 = TDIR@(ITHS);
 371     2     1                          DIR@1 = DIR@;
 372     2     1                          CALL WORD(VOC@,DIR@1,DIR@2); /* GO TO NEXT   */
                                                                       /*  LEVEL.      */
 373     2     1                          IF DKEY ¬= DINCOREKEY THEN         /* RESTORE */
 374     2     1                            DO; READ FILE(DRCTRY)            /* DIRBLOCK*/
                                                    INTO(DIRBLOCK) KEY(DKEY);
```

```
STMT LEVEL NEST

376     2    1                                          DINCOREKEY = DKEY;
377     2    1                                      END;
378     2    1                                    IF TKEY ¬= TINCOREKEY THEN          /* RESTORE */
379     2    1                                      DO; READ FILE(THES)               /* THSBLOCK*/
                                                          INTO(THSBLOCK) KEY(TKEY);
381     2    1                                          TINCOREKEY = TKEY;
382     2    1                                      END;
383     2    1                              END;
384     2    1    CATEND: END;  /* ENDS THE DO LOOP BEGUN AT THE STMT AFTER "CAT".      */
385     2                 LPATH = LPATH - 1;
386     2                 IF PRINTNDX > LPATH THEN PRINTNDX = LPATH;
                  /*******************************************************************/
388     2                 DIR@ = NEXTCAT;
389     2                 GO TO WORDRING;
390     2         WORDEND:LEVEL = LEVEL - 1;
391     2                 LPATH = LPATH - 1;
392     2                 IF PRINTNDX > LPATH-1 THEN PRINTNDX = LPATH-1;
394     2                 END WORD;

                  /*******************************************************************/
                  /*******************************************************************/

395     1         TEMPRNT: PROC(VKEY,IVOC);
                  /*******************************************************************/
                  /* CALLED BY "WORD".                                               */
                  /* FINDS AND PRINTS ALL TEMPORARY VOCAB ENTRIES WHOSE MATCNTS MATCH */
                  /* THAT OF THE VOCAB ENTRY AT IVOC.                                */
                  /* IF IVOC = -1 THEN THE CALL CAME FROM THE MAIN LOOP, AND ALL     */
                  /* NECESSARY INFORMATION IS IN THE KEY.                            */
                  /*******************************************************************/
396     2                 DCL   VKEY     FIXED DEC(5),    /* PARAMETER - VOCBLOCK KEY   */
                                IVOC     FIXED BIN;       /* PARAMETER -VOC ENTRY INDEX */
397     2                 DCL   VKEX     FIXED DEC(5);    /* LOCAL VOC BLOCK KEY        */
398     2                 DCL   VMC      FIXED BIN;       /* SAVE AREA FOR MATCNT       */
399     2                 DCL   BKT      CHAR(2);         /* TEMP STORE AREA FOR PAIR   */
400     2                 DCL   (IVK,JVK) FIXED BIN;      /* INDICES FOR VKEYS,VEXTS.   */
401     2                 DO I = (PRINTNDX+1) TO (LPATH-1);          /* PRINT THE WHOLE */
402     2    1              PUT EDIT(WORDSP(I))                      /* PATH WE HAVE    */
                                 (SKIP,COLUMN(N*I),A(LENGTH(WORDSP(I)))); /* BEEN   */
403     2    1              IF COUNTSP(I) ¬= -1 THEN PUT EDIT(COUNTSP(I))(F(5));
405     2    1              I = I + 1;                                     /* SAVING */
406     2    1              PUT EDIT(CATSP(I))(SKIP,COLUMN(N*I),A(8));
407     2    1            END;
408     2                 PRINTNDX = LPATH+1;
409     2                 P1 = ' '; P2 = ' ';               /* SCAN THE BUCKET.         */
411     2                 IF IVOC > -1              /* SAVE THE MATCNT & BUCKET DESIG, */
412     2                   THEN DO; VMC = VMATCNT(IVOC);     /* FOR WE MAY LOSE THE  */
414     2                            BKT = VWORD(IVOC);                   /* BLOCK.   */
415     2                        END;
416     2                   ELSE DO; VMC = KMATCNT;
418     2                            BKT = KWORD;
419     2                        END;
420     2                 IVK = UNSPEC(SUBSTR(BKT,1,1));         /* FETCH LAST BLOCK */
421     2                 JVK = UNSPEC(SUBSTR(BKT,2,1));         /* IN BUCKET.       */
```

```
STMT LEVEL NEST

422      2                IF IVK < 202 THEN IVK = IVK - 192;
424      2                  ELSE IF IVK < 218 THEN IVK = IVK - 199;
426      2                    ELSE IVK = IVK - 207;
427      2                IF JVK < 202 THEN JVK = JVK - 192;
429      2                  ELSE IF JVK < 218 THEN JVK = JVK - 199;
431      2                    ELSE JVK = JVK - 207;
432      2                VKEX = VKEYS(IVK,JVK)+VEXTS(IVK,JVK);
433      2                IF VKEX ¬= VINCOREKEY
434      2                  THEN DO; READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEX);
436      2                           VINCOREKEY = VKEX;
437      2                           K = 0;
438      2                       END;
439      2                  ELSE IF IVOC > -1
440      2                         THEN K = IVOC + 1;
441      2                         ELSE K = KVOCB - (VINCOREKEY*VBLKSIZE) + 1;
442      2        VSCAN3: DO IV = K TO (VBLKSIZE-1) WHILE(VWORD(IV)¬=' ');  /* SCAN FOR*/
443      2     1          IF (VFLAG(IV)>'2')&(VMATCNT(IV)=VMC) THEN  /* TEMP ENTRIES */
444      2     1            DO; PUT SKIP;                     /* WITH SAME MATCNT.     */
446      2     1              IF VSECT(IV) = TEXTSECT
447      2     1                THEN DO; IF IVOC = -1   /* SPECIAL DASH FOR LEVEL 0*/
449      2     1                           THEN CH1 = '-';
450      2     1                           ELSE DO; CH1 = ' ';
452      2     1                                    PUT EDIT(DASH)
                                                        (COLUMN(N*(LPATH-1)),A);
453      2     1                                END;
454      2     1                     END;
455      2     1                ELSE CH1 = ' ';
456      2     1              PUT EDIT(CH1,P1,VWORD(IV),P2)(COLUMN(N*LPATH),4 A);
457      2     1              IF VCOUNT(IV) ¬= -1 THEN PUT EDIT(VCOUNT(IV))(F(5));
459      2     1          END;
460      2     1        END;
461      2                IF VFLAG(VBLKSIZE-1) = '5' THEN     /* CHECK OVERFLOW BUCKET */
462      2                  DO; VKEX = VLASTBLK;
464      2                      READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEX);
465      2                      VINCOREKEY = VKEX;
466      2                      K = 0;
467      2                      GO TO VSCAN3;
468      2                  END;
469      2                IF VKEY ¬= VINCOREKEY THEN     /* RESTORE ORIG VOCAB BLOCK   */
470      2                  DO; READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEY);
472      2                      VINCOREKEY = VKEY;
473      2                  END;
474      2                END TEMPRNT;

                 /*********************************************************************/
                 /*********************************************************************/

475      1       PGHDG: PROCEDURE;
                 /* PAGE HEADING CONTROL -                                             */
476      2                DCL    HD11  CHAR(24)  INITIAL ('      -----------------'),
                               HD12  CHAR(16)  INITIAL ((16)'-'),
                               HD21  CHAR(24)  INITIAL ('      |   SEARCH KEY   |'),
                               HD22  CHAR(09)  INITIAL ('    LEVEL'),
                               HD23  CHAR(05)  INITIAL ('    |'),
STMT LEVEL NEST

                              HD31  CHAR(08) INITIAL ('       | '),
                              HD32  CHAR(16) INITIAL (' WORD  CATEGORY|');
 477     2                          PUT PAGE;
 478     2                    PGNO = PGNO + 1;
 479     2                    PUT EDIT('ANALYSIS ',KRQ#,'PAGE',PGNO) (A,F(5),X(97),A,F(4));
 480     2                          GO TO PH1;
 481     2           NOPHDG: ENTRY; /* ENTRY POINT FOR NOT TURNING PAGE.                 */
 482     2           PH1:               PUT SKIP;
 483     2                          IF N=4 THEN GO TO NOHDG;
 485     2                          PUT EDIT(HD11)(A);                    /* LINE 1  */
 486     2                          DO IHD=1 TO DEPTH; PUT EDIT(HD12)(A); END;
 489     2                          PUT EDIT(HD21)(SKIP,A);               /* LINE 2  */
 490     2                          DO IHD=1 TO DEPTH; PUT EDIT(HD22,IHD,HD23)
 492     2     1                      (A,F(2),A); END;
 493     2                          PUT EDIT(HD31)(SKIP,A);               /* LINE 3  */
 494     2                          DO IHD=1 TO DEPTH+1; PUT EDIT(HD32)(A); END;
 497     2                          PUT EDIT(HD11)(SKIP,A);               /* LINE 4  */
 498     2                          DO IHD=1 TO DEPTH; PUT EDIT(HD12)(A); END;
 501     2                          PUT SKIP;
 502     2                NOHDG: END PGHDG;

                 /**********************************************************************/
                 /**********************************************************************/

 503     1       CATSYP: PROCEDURE;
                 /**********************************************************************/
                 /* THIS SUBPROC PRINTS THE SYNOPSIS OF CATEGORIES APPEARING IN       */
                 /* "CATS", WHICH SHOULD BE ALL THOSE APPEARING IN A TREE.            */
                 /**********************************************************************/
                     /* FIRST WE SORT "CATS".                                          */
 504     2                DO I = 1 TO ICATSMAX-1;
 505     2     1            IF CATS(I) > CATS(I+1) THEN
 506     2     1              DO; DO J = I+1 TO 2 BY -1 WHILE(CATS(J)<CATS(J-1));
 508     2     2                    IX = CATS(J);
 509     2     2                    CATS(J) = CATS(J-1);
 510     2     2                    CATS(J-1) = IX;
 511     2     2                  END;
 512     2     1              END;
 513     2     1          END;
                     /* PAGE HEADING TRIVIA                                            */
 514     2                PGNO = PGNO + 1;
 515     2                PUT EDIT('ANALYSIS ',KRQ#,'PAGE',PGNO)
                                  (PAGE,A,F(5),X(97),A,F(4));
 516     2                PUT EDIT('THE FOLLOWING CATEGORIES APPEARED IN THIS SEARCH:')
                                  (SKIP(2),A);
 517     2                ON ENDPAGE(SYSPRINT)
 518     2                  BEGIN;  PGNO = PGNO + 1;
 520     3                          PUT EDIT('ANALYSIS ',KRQ#,'PAGE',PGNO)
                                            (PAGE,A,F(5),X(97),A,F(4));
 521     3                          PUT SKIP(2);
 522     3                  END;
                     /* NOW WE PRINT THE CATS, WITH THEIR WORDS.  WE DO NOT PRINT TEMP*/
                     /* VOCAB ENTRIES.                                                 */
 523     2                DO I = ICATSMAX TO 1 BY -1;
 STMT LEVEL NEST

 524     2    1                DKEY = CATS(I) / DBLKSIZE;
 525     2    1                IDIR = CATS(I) - (DKEY*DBLKSIZE);
 526     2    1                IF DKEY ¬= DINCOREKEY THEN
 527     2    1                  DO; READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEY);
 529     2    1                      DINCOREKEY = DKEY;
 530     2    1                  END;
 531     2    1                PUT EDIT(DCAT(IDIR),': ')(SKIP(2),A,A); /*PRINT CAT NAME   */
 532     2    1                CSW = 1;  PPOS = 109;
 534     2    1                TKEY = DTHS@(IDIR) / TBLKSIZE;
 535     2    1                ITHS = DTHS@(IDIR) - (TKEY*TBLKSIZE);
 536     2    1                IF TKEY ¬= TINCOREKEY THEN
 537     2    1                  DO; READ FILE(THES)INTO(THSBLOCK)KEY(TKEY);
 539     2    1                      TINCOREKEY = TKEY;
 540     2    1                  END;
 541     2    1                ITHS = ITHS - 1;
 542     2    1                DO J = 1 TO DLNG(IDIR);        /* PRINT EACH WORD IN THE CAT */
 543     2    2                  ITHS = ITHS + 1;
 544     2    2                  IF ITHS = TBLKSIZE THEN     /* HANDLE CAT OVERFLOW        */
 545     2    2                    DO; TKEY = TKEY + 1;  ITHS = 0;
 548     2    2                        READ FILE(THES)INTO(THSBLOCK)KEY(TKEY);
 549     2    2                        TINCOREKEY = TKEY;
 550     2    2                    END;
 551     2    2                  VKEY = TVOC@(ITHS) / VBLKSIZE;
 552     2    2                  IVOC = TVOC@(ITHS) - (VKEY*VBLKSIZE);
 553     2    2                  IF VKEY ¬= VINCOREKEY THEN
 554     2    2                    DO; READ FILE(VOCAB) INTO(VOCBLOCK)KEY(VKEY);
 556     2    2                        VINCOREKEY = VKEY;
 557     2    2                    END;
 558     2    2                  LWORD = LENGTH(VWORD(IVOC));     /* FIND ACTUAL LENGTH OF */
 559     2    2                  IF VWORD(IVOC)=' ' THEN                          /* WORD.    */
 560     2    2                    DO; LWORD = 0; GO TO C1; END;
 564     2    2                  DO WHILE(SUBSTR(VWORD(IVOC),LWORD,1) = ' ');   /* PEEL OFF*/
 565     2    2                    LWORD = LWORD - 1;                      /* EXCESS BLANKS   */
 566     2    2                  END;
 567     2    2    C1:           IF CSW = 1
 568     2    2                    THEN CSW = 0;
 569     2    2                    ELSE DO; PUT EDIT(',')(A);
 571     2    2                             PPOS = PPOS - 1;
 572     2    2                         END;
 573     2    2                  IF VCOUNT(IVOC)=-1 THEN LWORD = LWORD+2;  /*FOR ()        */
 575     2    2                  IF LWORD > PPOS THEN
 576     2    2                    DO; PPOS = 109;          /* NEW LINE                */
 578     2    2                        PUT EDIT((10)' ')(SKIP,A);
 579     2    2                    END;
 580     2    2                  PPOS = PPOS - LWORD;
 581     2    2                  IF VCOUNT(IVOC) = -1                   /* PRINT THE WORD   */
 582     2    2                    THEN PUT EDIT(' (',VWORD(IVOC),')')      /* NOT IN TEXT */
                                              (A,A(LWORD-2),A);
 583     2    2                    ELSE PUT EDIT(VWORD(IVOC))             /* IN TEXT     */
                                              (A(LWORD));
 584     2    2                END;
 585     2    1              END;
 586     2                   ICATS,ICATSMAX = 0;             /* RESET "CATS"              */
 587     2                   END CATSYP;
```

```
STMT LEVEL NEST

                  /*********************************************************************/
                  /*********************************************************************/
```

Sample Input and Output Data Sets for Previous Programs

```
*** THESAURUS ***

  THS@ ITHS          DIR@      VOC@        CATEGORY        WORD

** THS BLOCK          0
      0.    0.          3,        9        706.0           PROCESS
      1.    1.          4,       10                        PROGRAM
      2.    2.          2,        6        501.2           MIND
      3.    3.          0,        9                        PROCESS
      4.    4.          4,        3        501.1           MEMORY
      5.    5.          1,        6                        MIND
      6.    6.          4,        0        100.2           COMPUTER
      7.    7.          1,        9                        PROCESS
      8.    8.          0,       10                        PROGRAM
      9.    9.          3,        0        100.1           COMPUTER
     10.   10.          2,        3                        MEMORY
     11.   11.          3,       10                        PROGRAM
```

```
*** VOCABULARY ***

 VOC@ IVOC      MATCNT      SECT      DIR@     COUNT FLAG  TYPE

** VOC BLOCK       0
    0.    0.         -1,        -1,         4,        -1,     ,COMPUTER

** VOC BLOCK       1
    3.    0.         -1,        -1,         4,        -1,     ,MEMORY

** VOC BLOCK       2
    6.    0.         -1,        -1,         2,        -1,     ,MIND

** VOC BLOCK       3
    9.    0.         -1,        -1,         3,        -1,     ,PROCESS
   10.    1.         -1,        -1,         4,        -1,     ,PROGRAM

** VOC BLOCK       4
```

```
*** DIRECTOPY ***

  DIR@ IDIF   CATEGORY      THS@    LENGTH

** DIP BLOCK       (
     C.    0.  706.0    ,       0,         2
     1.    1.  501.2    ,       2,         2
     2.    2.  501.1    ,       4,         2
     3.    3.  100.2    ,       6,         3
     4.    4.  100.1    ,       9,         3
```

```
SAMPLE FOR ANNUAL REPORT        3/1/69

TEXT SECTION  1     MSGPARM IS 'LIST'

***** REQUEST ECIT

***SORTFD ANALYSIS REQUESTS TO BE PFOCESSED***

               ANALYSIS #  TYPE MODE THRESHOLD DEPTH KEYLIST CATEGCRY WCFI

  1.                    1    1    D       20      -1    LIST

  2.                    2    2    D       16      -1    LIST

  3.                    3    3    D        0      -1    LIST  100.2

  4.                    4    4    A        0      -1    LIST           MEMORY

  5.                    5    4    B        0      -1    LIST           MEMCRY

  6.                    6    4    C        0      -1    LIST           MEMORY

  7.                    7    4    D        0      -1    LIST           MEMCRY

  8.                    8    4    E        0      -1    LIST           MEMORY
```

```
***** VOCABULARY INITIALIZATION

***** THESAURUS UPDATE & SEARCH KEY GENERATION
         REQUEST# TYPE MODE THRESHOID MATCNT DEPTH    VOC@    DIR@ SECT VCOUNT VFLAG KFLAG CATEGORY WORD

                2    2    D         16      2  -1      3      4    1     20                         MEMORY
                2    2    D         16      4  -1      9      3    1     15    1                    PROCESS
                2    2    D         16      4  -1     11     -1    1      2    3                    PROCESSES

***** THESAURUS UPDATE COMPLETE

                1    1    D         20      0  -1     -1      1   -1     20               501.2
                1    1    D         20      0  -1     -1      2   -1     25               501.1
                1    1    D         20      0  -1     -1      4   -1     20               100.1
                3    3    D          0      0  -1     -1      3   -1     -1               100.2
                4    4    A          0      2  -1      3      4    1     20                         MEMORY
                5    4    B          0      2  -1      3      4    1     20                         MEMORY
                6    4    C          0      2  -1      3      4    1     20                         MEMORY
                7    4    D          0      2  -1      3      4    1     20                         MEMORY
                8    4    E          0      2  -1      3      4    1     20                         MEMORY

***** THESAUR NCRMAL TEPMINATION
        12 SEARCH KEYS GENERATED
        NC WARNING MESSAGES GENERATED
```

```
**********************************
ANALYSIS     1  -  MODE D,  TYPE 1.  SEARCH KEY IS 100.1    AND OCCURS  20 TIMES
                   SEARCH DEPTH = 3, EY DEFAULT
      ------------------------------------------------------------------
      |   SFARCH KEY   |    LEVFL 1    |    LEVEL 2    |    LEVEL 3    |
      | WCRD  CATEGORY| WORD  CATEGORY| WORD  CATFGORY| WORD  CATEGORY|
      ------------------------------------------------------------------


               100.1
                        (COMPUTER          )
                               100.2
                               --------  FROCESSES
                               --------  PROCFSS
                                               501.2
                                               --------  MIND
                                               706.0
                                        (PROGRAM          )
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
               -------   MEMORY
                               501.1
                               --------  MIND
                                               501.2
                                               --------  PROCESSES
                                               --------  PROCESS
                        (FRCGRAM           )
                               100.2
                               --------  PROCESSES
                               --------  FROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                               706.0
                               --------  PROCESSES
                               --------  FROCFSS
                                               100.2
                                               501.2
                                               --------  MIND
```

```
**********************************
ANALYSIS     1  -  MODE D, TYPE 1.  SEARCH KEY IS 501.1    AND OCCURS  25 TIMES
                   SEARCH DEPTH = 3, BY DEFAULT
      -----------------------------------------------------------------
      |   SEARCH KEY  |    LEVEL 1    |    LEVEL 2    |    LEVEL 3    |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      -----------------------------------------------------------------


               501.1
               --------  MEMORY
                               100.1
                                        (COMPUTER          )
                                               100.2
                                               --------  PROCESSES
                                               --------  PROCESS
                                        (PROGRAM           )
                                               100.2
                                               --------  PROCESSES
                                               --------  PROCESS
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
               --------  MIND
                               501.2
                               --------  PROCESSES
                               --------  PROCESS
                                               705.0
                                               100.2
```

```
**********************************
ANALYSIS     1  -  MODE D,  TYPE 1.  SEARCH KEY IS 501.2    AND OCCURS  20 TIMES
                   SEARCH DEPTH = 3, BY DEFAULT
      -----------------------------------------------------------------
      |   SEARCH KEY   |    LEVEL 1    |    LEVEL 2    |    LEVEL 3    |
      |  WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      -----------------------------------------------------------------


               501.2
               --------  MIND
                               501.1
                               --------  MEMORY
                                               100.1
               --------  PROCESSES
               --------  PROCESS
                               706.0
                                        (PROGRAM          )
                                               100.1 .
                                               --------  MEMORY
                                               100.2
                               100.2
                                        (COMPUTER         )
                                               100.1
                                               --------  MEMORY
                                        (PROGRAM          )
                                               100.1
                                               --------  MEMORY
```

```
***********************************
ANALYSIS     2  -  MODE D,  TYPE 2.  SEARCH KEY IS ROOT OF "MEMORY            " AND OCCURS AT LEAST  16 TIMES
                    SEARCH DEPTH = 3, BY DEFAULT
      --------------------------------------------------------------------
      |   SEARCH KEY    |    LEVEL 1     |    LEVEL 2     |    LEVEL 3     |
      |  WORD  CATEGORY|  WORD  CATEGORY|  WORD  CATEGORY|  WORD  CATEGORY|
      --------------------------------------------------------------------


         MEMORY
               100.1
                        (COMPUTER          )
                               100.2
                               --------  PROCESSES
                               --------  PROCESS
                                                100.2
                                                --------  MIND
                                                706.0
                                        (PROGRAM          )
                                                706.0
                                                ------    PROCESSES
                                                ------ -  PROCESS
                        (PROGRAM          )
                               100.2
                               --------  PROCESSES
                               --------  PROCESS
                                                501.2
                                                --------  MIND
                                                706.0
                               706.0
                               --------  PROCESSES
                               --------  PROCESS
                                                100.2
                                                501.2
                                                --------  MIND
               501.1
               --------  MIND
                               501.2
                               --------  PROCESSES
                               --------  PROCESS
                                                706.0
                                                100.2
```

```
**********************************
ANALYSIS     2  -  MODE D,  TYPE 2.  SEARCH KEY IS ROOT OF "PROCESS           " AND OCCURS AT LEAST  16 TIMES
                   SEARCH DEPTH = 3, BY DEFAULT
      -----------------------------------------------------------------
      |  SEARCH KEY   |    LEVEL 1    |    LEVEL 2    |    LEVEL 3    |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      -----------------------------------------------------------------


       . PROCESS
       - PROCESSES
               100.2
                        (COMPUTER          )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
                        (PROGRAM           )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
               501.2
               --------  MIND
                               501.1
                               --------  MEMORY
                                               100.1
               706.0
                        (PROGRAM           )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
                               100.2
                                        (COMPUTER          )
                                               100.1        '
                                               --------  MEMORY
```

```
**********************************
ANALYSIS     2  -  MODE D,  TYPE 2.  SEARCH KEY IS ROOT OF "PROCESSES          " AND OCCURS AT LEAST  16 TIMES
                   SEARCH DEPTH = 3, BY DEFAULT
      ------------------------------------------------------------------
      |  SEARCH KEY    |    LEVEL 1     |    LEVEL 2     |    LEVEL 3     |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ------------------------------------------------------------------

       . PROCESSES
       - PROCESS
               100.2
                        (COMPUTER          )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
                        (PROGRAM           )
                               100.1    .
                               --------  MEMORY
                                               501.1
                                               --------  MIND
               501.2
               --------  MIND
                               501.1
                               --------  MEMORY
                                               100.1
               706.0
                        (PROGRAM           )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
                               100.2
                                        (COMPUTER          )
                                               100.1
                                               --------  MEMORY
```

```
**********************************
ANALYSIS     3  -  MODE D,  TYPE 3.   SEARCH KEY IS 100.2
                   SEARCH DEPTH = 3, BY DEFAULT
      -------------------------------------------------------------------
      |   SEARCH KEY    |    LEVEL 1     |    LEVEL 2     |    LEVEL 3     |
      |  WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      -------------------------------------------------------------------


               100.2
                        (COMPUTER         )
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
                                        (PROGRAM          )
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
               --------  PROCESSES
               --------  PROCESS
                               501.2
                               --------  MIND
                                               501.1
                                               --------  MEMORY
                               706.0
                                        (PROGRAM          )
                                               100.1
                                               --------  MEMORY
                        (PROGRAM          )
                               706.0
                               --------  PROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                               100.1
                               --------  MEMORY
                                               501.1
                                               --------  MIND
```

```
***********************************
ANALYSIS     4  -  MODE A,  TYPE 4.  SEARCH KEY IS "MEMORY              "
                   SEARCH DEPTH = 3, BY DEFAULT
      ----------------------------------------------------------------------
      |  SEARCH KEY   |    LEVEL 1     |    LEVEL 2     |    LEVEL 3     |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ----------------------------------------------------------------------

       . MEMORY
               100.1
               501.1
               -------   MIND
                               501.2
                               --------  PROCESSES
                               --------  PROCESS
                                               706.0
                                               100.2
```

```
**********************************
ANALYSIS     5  -  MODE B,  TYPE 4.  SEARCH KEY IS "MEMORY            "
                   SEARCH DEPTH = 3, EY DEFAULT
      ------------------------------------------------------------------
      |   SEARCH KEY    |     LEVEL 1     |     LEVEL 2     |    LEVEL 3     |
      |  WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ------------------------------------------------------------------


         MEMORY
               100.1
                        (COMPUTER          )
                               100.2
                               --------  PROCESSES
                               --------  FROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                                        (PROGRAM           )
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
                        (FROGRAM           )
                               100.2
                               --------  FROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                               706.0
                               --------  PROCESSES
                               --------  FROCESS
                                               100.2
                                               501.2
                                               --------  MIND
               501.1
               --------  MIND
                               501.2
                               --------  PROCESSES
                               --------  FROCESS
                                               706.0
                                               100.2
```

```
**********************************
ANALYSIS     6  -  MODE C,  TYPE 4.  SEARCH KEY IS "MEMORY              "
                   SEARCH DEPTH = 3, BY DEFAULT
      ------------------------------------------------------------------
      |   SEARCH KEY   |    LEVEL 1    |    LEVEL 2    |    LEVEL 3    |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ------------------------------------------------------------------

         MEMORY
               100.1
                        (COMPUTER          )
                               100.2
                               --------  PROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                                                         (PROGRAM
                                        (PROGRAM            )
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
                        (PROGRAM           )
                               100.2
                                        (COMPUTER           )
                               --------  PROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                               706.0
                               --------  PROCESSES
                               --------  PROCESS
                                               100.2
                                                        (COMPUTER          )
                                               501.2
                                               --------  MIND
               501.1
               --------  MIND
                               501.2
                               --------  PROCESSES
                               --------  PROCESS
                                               706.0
                                                        (PROGRAM           )
                                               100.2
                                                        (COMPUTER          )
                                                        (PROGRAM           )
```

```
**********************************
ANALYSIS     7  -  MODE D,  TYPE 4.  SEARCH KEY IS "MEMORY           "
                   SEARCH DEPTH = 3, BY DEFAULT
      ---------------------------------------------------------------------
      |   SEARCH KEY   |    LEVEL 1     |    LEVEL 2     |    LEVEL 3     |
      |  WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ---------------------------------------------------------------------


       . MEMORY
               100.1
                        (COMPUTER          )
                               100.2
                               --------. PROCESSES
                               --------  PROCESS
                                         501.2
                                         --------  MIND
                                         706.0
                                        (PROGRAM          )
                                         706.0
                                         --------  PROCESSES
                                         --------  PROCESS
                        (PROGRAM           )
                               100.2
                               --------  PROCESSES
                               --------  PROCESS
                                         501.2
                                         --------  MIND
                                         706.0
                               706.0
                               --------  PROCESSES
                               --------  PROCESS
                                         100.2
                                         501.2
                                         --------  MIND
               501.1
               --------  MIND
                               501.2
                               --------  PROCESSES
                               --------  PROCESS
                                         706.0
                                         100.2
```

```
**********************************
ANALYSIS     8  -  MODE E,  TYPE 4.  SEARCH KEY IS "MEMORY
                   SEARCH DEPTH = 3, BY DEFAULT
      ------------------------------------------------------------------
      |  SEARCH KEY   |   LEVEL 1    |   LEVEL 2    |   LEVEL 3    |
      | WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY| WORD  CATEGORY|
      ------------------------------------------------------------------


         MEMORY
               100.1
                        (COMPUTER          )
                               100.2
                               --------  PROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                                                        (PROGRAM           )
                                        (PROGRAM           )
                                               706.0
                                               --------  PROCESSES
                                               --------  PROCESS
                        (PROGRAM           )
                               100.2
                                        (COMPUTER          )
                               --------  PROCESSES
                               --------  PROCESS
                                               501.2
                                               --------  MIND
                                               706.0
                               706.0
                               --------  PROCESSES
                               --------  PROCESS
                                               100.2
                                                        (COMPUTER          )
                                               501.2
                                               --------  MIND
              501.1
              --------  MIND
                              501.2
                              --------  PROCESSES
                              --------  PROCESS
                                              706.0
                                                        (PROGRAM           )
                                              100.2
                                                        (COMPUTER          )
                                                        (PROGRAM           )
```

APPENDIX C

Ring-Structure VIA Utility Programs

by

H. William Buttelmann

```
 1          PREVIA1:PROC(PARM)OPTIONS(MAIN);                                      PV10130
            /*******************************************************************/
            /* THIS IS THE FIRST OF TWO PROGRAMS (THE SECOND IS "PREVIA2") WHICH */
            /* CREATE THE FOUR DATA SETS COMPRISING A RING-STRUCTURED THESAURUS  */
            /* FROM A THESAURUS LISTING.  THE LISTING IS ACTUALLY SORTED IN INDEX*/
            /* FORMAT, AND IT IS THE INPUT DATA SET "ORGNX".                     */
            /*                                                                   */
            /* INPUT -1.CONTROL CARD WITH BLOCKLENGTHS IN BYTES FOR VOCABULARY,  */ PV10140
            /*          DIRECTORY, AND THESAURUS DATA SETS. BY MAKING THE BLOCK- */ PV10150
            /*          LENGTH AS LONG AS THE ENTIRE DATA SET, THE PROGRAM IS    */ PV10160
            /*          FORCED TO KEEP ALL THE DATA SET IN CORE.                 */ PV10170
            /*          EXAMPLE:                                                 */
            /*VBLKSIZE=2196, DBLKSIZE=1744, TBLKSIZE=7200;                       */
            /*        2.ORIGINAL INDEX; FORMAT: CATEGORY(8 BYTES) COMMA(1 BYTE)  */ PV10180
            /*                  WORD(18 BYTES). SORTED ON WORD,CATEGORY.         */ PV10190
            /*                  MAY BE ENTERED THRU EITHER "SYSIN" OR "ORGNX"    */
            /*                  FILE; PARM IN EXEC CARD MUST SPECIFY WHICH;      */
            /*                  DEFAULT FOR THE PARM IS "ORGNX".                 */
            /*        3.PARM.GO='SYSIN'                                          */
            /*             OR                                                    */
            /*          PARM.GO='ORGNX'             (DEFAULT)                    */
            /* OUTPUT-1.INDEX; FORMAT: CATEGORY(8 BYTES) VOCABULARY POINTER(1 WD)*/ PV10210
            /*        2.VOCABULARY DATA SET PORTION OF THESAURUS. DIRECT ACCESS  */ PV10220
            /*          REGIONAL(1). ORGANIZATION IS IN BUCKETS, KEYED BY A      */ PV10230
            /*                 KEY XFORMATION(TABLE LOOKUP) BASED ON INITIAL     */ PV10240
            /*                 DIGRAPH OF WORD. BUCKETS HAVE SPACE FOR TEMPORARY*/ PV10250
            /*                 ADDITIONS. THE FINAL BUCKET IS AVAILABLE FOR      */ PV10260
            /*                 OVERFLOW.                                         */ PV10270
            /*        3.THSCTL: TABLE OF CONTROL INFO FOR 3 THESAURUS DATA SETS. */ PV10280
            /*                 INCLUDES BLOCK LENGTHS(IN # OF RECORDS), VKEY     */ PV10290
            /*                 AND VEXT ARRAYS.                                  */ PV10300
            /*******************************************************************/
 2                 DCL   PARM  CHAR(1);  /* SEE PARM.GO - INPUT ITEM 3 IN        */
                                                           /* DOCUMENTATION.     */
 3                 DCL   SYSIN  FILE STREAM INPUT,                                PV11010
                         ORGNX  FILE STREAM INPUT,   /* INPUT NDX, IF NOT SYSIN.*/ PV11015
                         INDEX  FILE RECORD OUTPUT,                               PV11020
                         VOCAB  FILE RECORD OUTPUT KEYED SEQUENTIAL               PV11030
                                  ENVIRONMENT(REGIONAL(1)),                       PV11040
                         THSCTL FILE STREAM OUTPUT;                               PV11050
 4                 DCL   VBLKSIZE         FIXED BIN,      /* BLOCK SIZES:   MUST BE*/ PV11060
                         DBLKSIZE         FIXED BIN,      /*    1ST CARD IN SYSIN  */ PV11070
                         TBLKSIZE         FIXED BIN,                                PV11080
                         VKEY(26,26)      FIXED DEC(3),   /* KEY OF 1ST BLK IN     */ PV11090
                                                          /*    BUCKET DEFINED BY  */ PV11100
                                                          /*    LEADING DIGRAPH    */ PV11110
                         VEXT(26,26)      FIXED DEC(3),   /* EXTRA BLOCKS IN BUCKET*/ PV11120
                         BLKCNT           FIXED DEC(3) INITIAL(-1),                 PV11130
                         CAT              CHAR(8),        /* INPUT INDEX ITEM      */ PV11140
                         WORD             CHAR(18),       /* INPUT INDEX ITEM      */ PV11150
                         WORDSAV          CHAR(18),                                 PV11160
                         VOC@             FIXED BIN,      /* VOCABULARY ENTRY INDEX*/ PV11170
                         VLASTBLK         FIXED BIN,                                PV11180
```

```
                       (I,J)             FIXED BIN(8);  /* VKEY & VEXT INDICES    */ PV11190
 5               DCL  01 VOCBLOCK CONTROLLED,           /* VOCAB ENTRY            */ PV11200
                        02 VOCRCD(VBLKSIZE),            /* VBLKING IS CONVERTED   */ PV11210
                          03 VM   FIXED BIN,            /* FROM BYTES TO RCDS.    */ PV11240
                          03 VS   FIXED BIN,                                         PV11250
                          03 VD   FIXED BIN,                                         PV11260
                          03 VC   FIXED BIN,                                         PV11270
                          03 VX   CHAR(1),                                           PV11275
                          03 VF   CHAR(1),                                           PV11280
                          03 VW   CHAR(18);                                          PV11290
 6               DCL  01 INDRCD,                        /* INDEX OUTPUT RECORD    */ PV11300
                        02 ICAT  CHAR(8),               /* CATEGORY               */ PV11310
                        02 IVOCB FIXED BIN,             /* VOCABULARY POINTER     */ PV11320
                        02 IFILL CHAR(6);               /* FOR MIN LRECL 18 BYTES*/ PV11330
 7               DCL  FIL CHAR(1);          /* FOR READING CARD IMAGE RCDS        */ PV11340
 8               DCL  REJSW FIXED BIN        /* REJECT SWITCH                     */ PV11350
                               INITIAL (0);
 9       BEGIN:  ON ENDFILE(ORGNX) GO TO LASTRCD;                                    PV13000
11               ON ENDFILE(SYSIN) GO TO LASTRCD;
13               GET DATA (VBLKSIZE,DBLKSIZE,TBLKSIZE);/* READ THE CONTROL CD,*/ PV13020
14               PUT SKIP; PUT DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE); /* PRINT IT, */ PV13021
16               VBLKSIZE = VBLKSIZE / 36;               /* & CONVERT BLKINGS TO*/ PV13030
17               DBLKSIZE = DBLKSIZE / 16;               /* # OF RCDS PER BLK   */ PV13040
18               TBLKSIZE = TBLKSIZE / 08;                .                          PV13050
19               ALLOCATE VOCBLOCK;                    /* INITIALIZE VOC BLOCK  */ PV13060
20               DO IVOC = 1 TO VBLKSIZE;                                            PV13070
21                 VM(IVOC) = -1;                                                    PV13090
22                 VS(IVOC) = -1;                                                    PV13100
23                 VD(IVOC) = -1;                                                    PV13110
24                 VC(IVOC) = -1;                                                    PV13120
25                 VX(IVOC) = ' ';                                                   PV13125
26                 VF(IVOC) = ' ';                                                   PV13130
27               END;                                                                PV13140
28               VKEY = -1; VEXT = -1; VOCB = -1;
31               IF PARM='S'
32                 THEN GET FILE(SYSIN)EDIT(CAT,WORD,FIL)
                         (A(8),X(1),A(18),X(52),A(1));
33                 ELSE GET FILE(ORGNX)EDIT(CAT,WORD,FIL)
                         (A(8),X(1),A(18),X(52),A(1));
34               GO TO RESET;                                                        PV13170
35      GETITEM: IF PARM='S'
36                 THEN GET FILE(SYSIN)EDIT(CAT,WORD,FIL)
                         (A(8),X(1),A(18),X(52),A(1));
37                 ELSE GET FILE(ORGNX)EDIT(CAT,WORD,FIL)
                         (A(8),X(1),A(18),X(52),A(1));
38               IF REJSW = 1                                                        PV13190
39                 THEN IF WORD = WORDSAV                                            PV13200
40                         THEN GO TO GETITEM;                                       PV13210
41                         ELSE DO; REJSW=0; GO TO RESET; END;                       PV13220
45               IF WORD = WORDSAV THEN GO TO PUTITEM;                               PV13230
47               IF WORD < WORDSAV THEN   /* SEQUENCE CHECK - ABORT IF OUT    */ PV13240
48                 DO; PUT EDIT('**** INPUT INDEX OUT OF SEQUENCE!! ',               PV13250
                              'LAST GOOD ENTRY:''',WORDSAV,'''. RUN ABORTED.') PV13260
```

```
                                (SKIP(2),A,A,A(18),A);                            PV13270
                        I = I / 0;      /* CAUSE ABNORMAL END                  */ PV13280
                   END;                                                           PV13290
                 IF (SUBSTR(WORD,1,2)¬=SUBSTR(WORDSAV,1,2))|(IVOC=VBLKSIZE) THEN PV13300
                   DO; BLKCNT = BLKCNT + 1;                                       PV13310
                        VOC@  = (BLKCNT+1)*VBLKSIZE - 1;                          PV13320
                        IF VKEY(I,J) = -1 THEN VKEY(I,J) = BLKCNT;                PV13330
                        VEXT(I,J) = VEXT(I,J) + 1;                                PV13340
                        WRITE FILE(VOCAB)FROM(VOCBLOCK)KEYFROM('    '||BLKCNT);   PV13390
        RESET:          DO IVOC = 1 TO VBLKSIZE; VW(IVOC) = (18)' '; END;        PV13400
                        IVOC = 0;                                                 PV13410
        CI:             I = UNSPEC(SUBSTR(WORD,1,1)); /* COMPUTE BUCKET INDICES*/ PV13420
                        IF I < 202 THEN DO; I = I-192; GO TO CJ; END;             PV13430
                        IF I < 218 THEN DO; I = I-199; GO TO CJ; END;             PV13440
                        I = I-207;                                                PV13450
        CJ:             J = UNSPEC(SUBSTR(WORD,2,1));                             PV13460
                        IF J < 202 THEN DO; J = J-192; GO TO FX; END;             PV13470
                        IF J < 218 THEN DO; J = J-199; GO TO FX; END;             PV13480
                        J = J-207;                                                PV13490
        FX:             IF (I<1)|(I>26)|(J<1)|(J>26) THEN /* TEST FOR PROPER   */ PV13500
                          DO; REJSW = 1;                  /* SUBSCRIPT RANGE   */ PV13510
                              PUT EDIT('****WORD ''',WORD,''' REJECTED.')         PV13520
                                        (SKIP,A,A(18),A);                         PV13530
                              WORDSAV = WORD;                                     PV13540
                              GO TO GETITEM;                                      PV13550
                          END;                                                    PV13560
                   END;                                                           PV13570
                 WORDSAV = WORD;                                                  PV13580
                 IVOC = IVOC + 1; VOC@ = VOC@ + 1;                                PV13590
                 VW(IVOC) = WORD;                                                 PV13600
       PUTITEM: ICAT = CAT;                                                       PV13610
                 IVOC@ = VOC@;                                                    PV13620
                 IFILL = ' ';                                                     PV13630
                 WRITE FILE(INDEX) FROM(INDRCD);
                 GO TO GETITEM;
       LASTRCD: IF REJSW = 1 THEN GO TO OVB;
                 BLKCNT = BLKCNT + 1;                     /* WRITE LAST BLK OF DATA*/ PV13680
                 IF VKEY(I,J) = -1 THEN VKEY(I,J) = BLKCNT;                       PV13690
                 VEXT(I,J) = VEXT(I,J) + 1;                                       PV13700
                 WRITE FILE(VOCAB)FROM(VOCBLOCK)KEYFROM('    '||BLKCNT);          PV13760
       OVB:      BLKCNT = BLKCNT + 1;                     /* WRITE OVERFLOW BLOCK  */ PV13770
                 DO IVOC = 1 TO VBLKSIZE; VW(IVOC) = (18)' '; END;                PV13780
                 WRITE FILE(VOCAB)FROM(VOCBLOCK)KEYFROM('    '||BLKCNT);          PV13790
                 DO I = 1 TO 26; DO J = 1 TO 26;          /* ASSIGN ALL UNUSED     */ PV13800
                   IF VEXT(I,J) = -1 THEN                 /* BUCKETS TO OVERFLOW   */ PV13810
                     DO; VKEY(I,J) = BLKCNT;              /* BLOCK.                */ PV13820
                         VEXT(I,J) = 0;                                           PV13830
                     END;                                                         PV13840
                 END; END;                                                        PV13850
                 VLASTBLK = BLKCNT;                                               PV13860
                 PUT FILE(THSCTL)                         /* CREATE DATA SET OF    */ PV13870
                      DATA(VBLKSIZE,DBLKSIZE,             /*   CONTROL INFO FOR    */ PV13880
                           TBLKSIZE,VLASTBLK);            /*   THESAURUS DATA SETS */ PV13890
       PREVIA1:PROC(PARM)OPTIONS(MAIN);                                 PV10130            PAGE     5

128                PUT FILE(THSCTL)EDIT(VKEY,VPXT)(F(4));                        PV13900
129           END PREVIA1;                                                       PV13930
```

```
          PREVIA2: PROC OPTIONS(MAIN);
          /*********************************************************************/
          /* THIS PROGRAM IS THE SECOND HALF OF THE STANDARD PREVIA PACKAGE.   */
          /* SEE "PREVIA1" FOR FIRST HALF AND FOR INITIAL DOCUMENTATION.       */
          /* PREVIA2 SCANS THE INPUT THESAURUS, NOW SORTED INTO THESAURUS      */
          /* ORDER, AND THE VOCAB PARTIALLY BUILT BY "PREVIA1", AND BUILDS THE */
          /* REMAINDER OF THE DATA SETS COMPRISING THE RING THESAURUS,         */
          /* "DRCTRY" AND "THES", AND FINALLY COMPLETES THE "VOCAB" DATA.SET.  */
          /*********************************************************************/
               DCL  VOCAB  FILE RECORD DIRECT UPDATE                                PV30200
                                ENVIRONMENT(REGIONAL(1)),                           PV30210
                    DRCTRY FILE RECORD KEYED                                        PV30220
                                ENVIRONMENT(REGIONAL(1)),                           PV30230
                    THES   FILE RECORD KEYED                                        PV30240
                                ENVIRONMENT(REGIONAL(1)),                           PV30250
                    INTHS  FILE INPUT RECORD,                                       PV30260
                    THSCTL FILE       STREAM;                                       PV30270
               DCL  VBLKSIZE   FIXED BIN,      /* # RECORDS PER BLOCK         */ PV30280
                    DBLKSIZE   FIXED BIN,                                           PV30290
                    TBLKSIZE   FIXED BIN,                                           PV30310
                    VLASTBLK   FIXED BIN,      /* # OF LAST BLOCK IN VOC -    */ PV30320
                                               /*   THE OVERFOLW BUCKET.      */ PV30330
                    DLASTBLK   FIXED BIN,      /* # OF LAST BLOCK IN DIR      */ PV30340
                    TLASTBLK   FIXED BIN;      /* # OF LAST BLOCK IN THS      */ PV30350
               DCL  VKEY       FIXED DEC(5),   /* VOCABULARY BLOCK KEY        */ PV30360
                    DKEY       FIXED DEC(5),   /* DIRECTORY  BLOCK KEY        */ PV30370
                    TKEY       FIXED DEC(5),   /* THESAURUS  BLOCK KEY        */ PV30380
                    VOC@       FIXED BIN,      /* VOCABULARY INDEX: 0-ORIGIN */ PV30382
                    DIR@       FIXED BIN,      /* DIRECTORY  INDEX: 0-ORIGIN */ PV30390
                    THS@       FIXED BIN,      /* THESAURUS  INDEX: 0-ORIGIN */ PV30400
                    VINCOREKEY FIXED DEC(5),   /* VOC BLOCK IN CORE           */ PV30410
                    DINCOREKEY FIXED DEC(5),   /* DIR BLOCK IN CORE           */ PV30420
                    TINCOREKEY FIXED DEC(5);   /* THS BLOCK IN CORE           */ PV30430
               DCL  01 INTHSREC,               /* INPUT THESAURUS - SORTED    */ PV30440
                      02 ITCAT CHAR(8),        /*   OUTPUT FROM PREVIA1       */ PV30450
                      02 ITV@  FIXED BIN,                                           PV30460
                      02 IFILL CHAR(6);        /* TO GIVE MIN LRECL 18 BYTES */ PV30470
               DCL  01 VOCBLOCK CONTROLLED,    /* VOCABULARY ENTRY            */ PV30490
                      02 VOCRCD(0:VBLKSIZE-1),                                      PV30500
                        03 VM        FIXED BIN,                                     PV30530
                        03 VS        FIXED BIN,                                     PV30540
                        03 VDIR@     FIXED BIN,                                     PV30550
                        03 VLINK     FIXED BIN,                                     PV30560
                        03 VX        CHAR(1),                                       PV30565
                        03 VF        CHAR(1),                                       PV30570
                        03 VW        CHAR(18);                                      PV30580
               DCL  01 THSBLOCK CONTROLLED,    /* THESAURUS DATA SET.         */ PV30590
                      02 THSRCD(0:TBLKSIZE-1),/*   WE DO OUR OWN BLOCKING.    */ PV30600
                        03 TDIR@     FIXED BIN,                                     PV30610
                        03 TVOC@     FIXED BIN;                                     PV30620
               DCL  01 DIRBLOCK CONTROLLED,    /* DIRECTORY DATA SET.         */ PV30630
                      02 DIRRCD(0:DBLKSIZE-1),/*   WE DO OUR OWN BLOCKING.    */ PV30640
                        03 DCAT      CHAR(8),                                       PV30650
```

```
                          03 DTHS@     FIXED BIN,                                          PV30660
                          03 CLNG      FIXED BIN;                                          PV30670
  9           DCL CATSAV        CHAR(8),          /* USPD IN CATFGOFY                 */ PV30680
                  THS@SAV       FIXED BIN;        /*   BREAK.                         */ PV30690
 10           DCL  CATLNG       FIXED BIN INITIAL (0);   /* CATEGORY LENGTH           */ PV30700
 11           DCL  SW1          FIXED BIN INITIAL (1);                                    PV30710
 12   BEGIN:  ON ENDFILE(INTHS, GO TO LASTPCD;                                            PV33000
 14           OPEN FILE(THSCTL) INPUT;                                                    PV33010
 15           GET FILE(THSCTL)DATA(VPLKSIZP,DBIKSIZE,TBIKSIZE,VLASTBLK);                  PV33020
 16           ALLOCATE VOCBLOCK, THSBLOCK, DIRBLOCK;                                      PV33030
 17           IDIR, ITHS, DIR@, THS@ = 0;                                                 PV33040
 18           DKEY, TKEY = 0;                                                             PV33050
 19           VINCOREKEY, DINCOREKEY, IINCOREKEY = -1;                                    PV33060
 20           OPEN FILE(DRCTRY) SEQUENTIAL OUTPUT,                                        PV33070
                   FILE(THES)   SEQUENTIAL OUTPUT;                                        PV33080
 21           READ FILE(INTHS)INTO(INTHSRCD);                                             PV33085
 22           CATSAV = ITCAT; THS@SAV = THS@;                                             PV33086
 24           GO TO READVOC;                                                              PV33087
 25   READTHS:READ FILE(INTHS)INTO(INTHSRCD);                                             PV33090
 26           ITHS = ITHS + 1; THS@ = THS@ + 1;                                           PV33100
 28   READVOC:VKEY = ITV@ / VBLKSIZE;           /* COMPUTE VCC KEY @              */ PV33120
 29           IVOC = ITV@ - (VBLKSIZE*VKEY); /*   POSN WITHIN BLOCK.            */ PV33130
 30           IF VKEY = VINCOREKEY              /* IF VOC BICCK ALREADY IN        */ PV33140
                THEN GC TO CATBRK;              /*   CORE, BYPASS FETCH.          */ PV33150
 31           IF SW1 = 1                                                                  PV33160
 32             THEN SW1 = 0;                                                             PV33170
 33             ELSE REWRITE FILE(VOCAB)FROM(VOCBLOCK)KEY(VINCOREKEY);                    PV33180
 34
 35           READ FILE(VOCAB)INTO(VOCBLOCK) KEY(VKEY); /*EVENT(VREAD); IN- */ PV33190
                                                        /* CLUDE THE EVENT  */ PV33200
                                                        /* CPTICN WHEN IT   */ PV33210
                                                        /* IS SUPPORTED.    */ PV33220
 36           VINCOREKEY = VKEY;                                                          PV33230
 37   CATBRK: IF ITCAT = CATSAV THEN GO TO INITVDIR@;                                     PV33240
 39           DCAT(IDIR) = CATSAV;                                                        PV33250
 40           DTHS@(IDIR) = THS@SAV;                                                      PV33260
 41           DLNG(IDIR) = CATLNG;                                                        PV33270
 42           CATSAV = ITCAT; THS@SAV = THS@; CATLNG = 0;                                 PV33280
 45           IDIR = IDIR + 1; DIR@ = DIR@ + 1;                                           PV33282
 47           IF IDIR > DBLKSIZE-1 THEN                                                   PV33290
 48             DO; WRITE FILE(DRCTRY)FROM(DIRBLOCK)KEYFRCM(DKEY);                        PV33300
 50                 DKEY = DKEY + 1;                                                      PV33310
 51                 IDIR = -1;                                                            PV33320
 52             END;                                                                      PV33330
 53   INITVDIR@: /* WAIT(VREAD); - INCLUDE THE EVENT OPTICN WHEN IT IS            */ PV33340
                                                              /* SUPPORTED.       */ PV33350
              IF VDIR@(IVOC) = -1 THEN         /* 1ST TIME SAVE DIR@ IN           */ PV33360
 54             DO; VLINK(IVOC) = DIR@;        /*   VLINK FOR CLOSING RING,       */ PV33370
 56                 VDIR@(IVOC) = DIR@;        /* & IN VDIR@ IN CASE THIS IS      */ PV33375
 57             END;                           /* A SINGLE ENTRY.                 */ PV33380
 58   THSENTRY: TDIR@(ITHS) = VDIR@(IVOC);     /* PRODUCE THESAURUS ENTRY         */ PV33390
 59           TVOC@(ITHS) = ITV@;              /*   USE DIR@ IN VDIR@ TO          */ PV33400
                                               /*   CHAIN BACKWARDS.              */ PV33401
 60           IF ITHS = TBLKSIZE-1 THEN                                                   PV33410
 61                DO: WRITE FILE(THES)FROM(THSBLOCK)KEYFROM(TKEY);                        PV33420
 63                    TKEY = TKEY + 1;                                                    PV33430
 64                    ITHS = -1;                                                          PV33440
 65                END;                                                                    PV33450
 66       VOCLINK:VDIR@(IVOC) = DIR@;               /* SAVE DIR@ IN VDIR@ FOR       */ PV33460
                                                    /*   CHAINING.                  */ PV33461
 67               CATLNG = CATLNG + 1;              /* COUNT IN CATEGORY LENGTH     */ PV33462
 68               GO TO READTHS;                    /* END OF MAIN PROGRAM          */ PV33470
 69       LASTRCD:REWRITE FILE(VOCAB)FROM(VOCBLOCK)KEY(VINCOREKEY);                      PV33480
 70               DCAT(IDIR) = CATSAV;              /* FINISH OUT LAST CATBRK.      */ PV33482
 71               DTHS@(IDIR) = THS@SAV;                                                 PV33483
 72               DLNG(IDIR) = CATLNG;                                                   PV33484
 73               DO IDIR = IDIR+1 TO DBLKSIZE-1;/* PAD OUT AND WRITE LAST          */ PV33490
 74                 DCAT(IDIR) = (8)'9';            /*   DIRECTORY BLOCK.           */ PV33500
 75                 DTHS@(IDIR) = THS@ + 1;                                              PV33510
 76               DLNG(IDIR) = 0;                                                        PV33512
 77               END;                                                                   PV33520
 78               WRITE FILE(DRCTRY)FROM(DIRBLOCK)KEYFROM(DKEY);                         PV33530
 79               DO ITHS = ITHS+1 TO TBLKSIZE-1;/* PAD OUT AND WRITE LAST          */ PV33540
 80                 TDIR@(ITHS) = -1;               /*   THESAURUS BLOCK            */ PV33550
 81                 TVOC@(ITHS) = -1;                                                    PV33560
 82               END;                                                                   PV33570
 83               WRITE FILE(THES)FROM(THSBLOCK)KEYFROM(TKEY);                           PV33580
 84               DLASTBLK = DKEY; TLASTBLK = TKEY;                                      PV33590
 85               SW1 = 1;                                                               PV33600
 87       FINALPASS:                                /* REOPEN THE DIRECTORY &       */ PV33610
                  CLOSE FILE(DRCTRY),               /* THESAURUS DATA SETS FOR      */ PV33620
                        FILE(THES);                 /* DIRECT UPDATING.             */ PV33630
 88               OPEN FILE(DRCTRY)DIRECT INPUT,                                         PV33640
                       FILE(THES)  DIRECT UPDATE;                                        PV33650
 89               DO VKEY = 0 TO VLASTBLK;          /* READ THE VOC SEQUENTIALLY.  */ PV33660
 90                 READ FILE(VOCAB)                /* FIND THE 1ST THS ENTRY FOR  */ PV33670
                         INTO(VOCBLOCK)             /*   EACH WORD.                */ PV33680
                         KEY(VKEY);                 /* LINK ITS DIR@ TO THE LAST   */ PV33690
                                                    /*   THS ENTRY FOR THE WORD,   */ PV33700
                                                    /*   THUS CLOSING THE WORD.    */ PV33710
                                                    /*   RINGS.                    */ PV33720
 91                 DO IVOC = 0 TO VBLKSIZE-1 WHILE(VW(IVOC)¬=(18)' ');                  PV33730
 92                     IF VLINK(IVOC) = VDIR@(IVOC) /* IF THIS WORD HAS ONLY 1*/ PV33732
 93                        THEN GO TO VPRSTOR;       /* TOKEN IN THES, THERE IS*/ PV33733
                                                     /* NO CLOSING TO DO.      */ PV33734
 94                     DKEY = VLINK(IVOC) / DBLKSIZE;                                   PV33740
 95                     IDIR = VLINK(IVOC) - (DKEY*DBLKSIZE);                            PV33750
 96                     IF DKEY ¬= DINCOREKEY THEN                                       PV33760
 97                       DO; READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEY);                  PV33770
 98                           DINCOREKEY = DKEY;                                         PV33780
100                       END;                                                           PV33790
101                     TKEY = DTHS@(IDIR) / TBLKSIZE;                                   PV33800
102                     ITHS = DTHS@(IDIR) - (TKEY*TBLKSIZE);                            PV33810
103                     VOC@ = (VKEY*VBLKSIZE) + IVOC;                                   PV33812
104                     IF TKEY ¬= TINCOREKEY THEN                                       PV33820
105                       DO; IF SW1 = 1                                                 PV33830
107                              THEN SW1 = 0;                                           PV33840
```

```
                      ELSE REWPITE FILE(THES)FROM(TESBLOCK)                 PV33850
                                           .  KEY(TIKCOREKEY);              PV33860
                    READ FILE(THES)INTO(THSBLOCK)KEY(TKEY);                 PV33870
                    TINCOREKEY = TKEY;                                      PV33880
                END;                                                        PV33890
              ITHS = ITHS - 1;                                              PV33892
              DO I = 1 TO DLNG(IDIR);                                       PV33893
                ITHS = ITHS + 1;                                            PV33894
                IF ITHS = TBLKSIZE THEN                                     PV33895
                  DO: REWRITE FILE(THES)FROM(THSBLOCK)KEY(TINCOREKEY);      PV33896
                      TKEY = TKEY + 1; ITHS = 0;                            PV33897
                      READ FIIF(THES)INTO(THSBLOCK)KEY(TKEY);               PV33898
                      TINCOREKEY = TKEY;                                    PV33899
                  END;                                                      PV33900
                IF TVOC@(ITHS) = VOC@ THEN GO TO VFCUND;                    PV33901
              END;                                             -            PV33902
    VNOTFOUND:PUT EDIT('!! VOC ENTRY NOT FOUND IN THES IN FINALPASS.',      PV33903
                       ' RUN ABORTED.')(SKIP(2),A,A);                       PV33904
              PUT SKIP;PUT DATA(VKEY,IVOC,DKEY,IDIR,TKEY,ITHS,VOC@);        PV33905
              REWRITE FILF(VOCAB)FROM(VOCBLOCK)KEY(VKFY);                   PV33906
              FEWRITE FILF(THES)FROM(THSBLOCK)KEY(TINCOREKEY);              PV33906
              GO TO UPDATETHSCTL;                                           PV33907
    VFCUND:   TDIR@(ITHS) = VDIR@(IVOC);     /* CLOSE RING.              */ PV33908
    VFESTOR:  VLINK(IVOC) = -1;              /* RESTCRE VLINK.           */ PV33909
          END;                                                              PV33910
          REWRITE FILE(VOCAB)FROM(VOCBLOCK)KEY(VKEY);                       PV33915
        END;                                                                PV33920
        REWRITE FILE(THFS)FROM(THSBLOCK)KEY(TINCOREKEY);                    PV33930
UPDATFTHSCTL:  FREE VOCBLOCK, THSBLOCK, DIRBLOCK; /* GET SOME SPACE      */ PV33940
        BEGIN;                                                              PV33950
          DCL (VKEY(26,26),VEXT(26,26)) FIXED DEC(3) CONTROLLED;            PV33960
          ALLOCATE VKEY, VEXT;                                              PV33970
          GET FILE(THSCTL)EDIT(VKEY,VEXT)(F(4));                            PV33980
          CLOSE FILE(THSCTL);                                               PV33990
          OPEN  FILE(THSCTL) CUTPUT;                                        PV34000
          PUT FILE(THSCTL)DATA(VBLKSIZE,DBLKSIZE,TBIKSIZE,                  PV34010
                               VLASTBLK,DLASTBLK,TIASTBLK);                 PV34020
          PUT FILE(THSCTL)EDIT(VKEY,VEXT)(F(4));                            PV34030
        PUT EDIT('BLKSIZES IN # OF RECORDS:')(A);
          PUT SKIP; PUT DATA(VBLKSIZE,DBLKSIZE,TBIKSIZE,                    PV34032
                           VLASTBLK,DLASTBLK,TLASTBIK);                     PV34033
        END;                                                                PV34040
ENDPV3: END PREVIA2;
```

```
MAKTEXT: PROC(PARM)OPTIONS(MAIN);
/***************************************************************************/
/* CREATES TEXT FILE INPUT TO "THESAUR" FROM CARD INPUT.                   */
/* INPUT & OUTPUT RECORDS HAVE FORMAT IDENTICAL TO THE OUTPUT FROM         */
/* THE PROGRAM "SUFFIX", EXCEPT THAT THE CARD INPUT TO THIS PROGRAM        */
/* HAS ONE RECORD PER CARD.                                                */
/* PARM.GO='LIST' IN THE EXEC CARD CAUSES A LISTING OF THE OUTPUT          */
/* RECORDS.                                                                */
/***************************************************************************/
        DCL PARM CHAR(1);   /* EXEC CARD PARAMETER.                        */
        DCL TEXT FILE STREAM OUTPUT;   /* OUTPUT TEXT FOR "THESAUR"        */
        DCL TEXTRCD  CHAR(30);         /* INPUT FIELD - ONLY FIRST 30*/
                                       /* CHARACTERS ARE READ.             */
        ON ENDFILE(SYSIN) GO TO ENDMAKTEXT;
GETT:   GET FILE(SYSIN) EDIT(TEXTRCD) (A(80));
        PUT FILE(TEXT) EDIT(TEXTRCD) (A(30));
        IF PARM='L' THEN PUT EDIT(TEXTRCD) (SKIP,A);
        GO TO GETT;
ENDMAKTEXT: END;
```

```
 1              THSPRNT:PROC OPTIONS(MAIN);                                            THP0140
                /***********************************************************************/ THP0142
                /* THIS PROCEDURE PRINTS THE THREE DATA SETS COMPFISING THE           */ THP0150
                /* RING-STRUCTURED THESAURUS USED IN VIA.  THE DATA SETS ARE:          */ THP0160
                /* VCCABULARY, DIRECTORY, THESAURUS.                                   */ THP0170
                /* IT WILL ALSC PRINT THE CONTROL DATA FOR THE THESAURUS IN THE        */
                /* DATA SET, THSCTL.                                                   */
                /* CHOICE OF DATA SETS TO BE FRINTED IS CONTROLLED PY CONTROL CARDS    */ THP0180
                /* ENTERED IN SYSIN.  ONLY DATA SETS SPECIFICALLY FEQUESTED ARE        */ THP0190
                /* PRINTED.  CONTROL CARD FORMAT IS DATA-DIRECTED.                     */ THP0200
                /* EXAMPLES:                                                           */ THP0210
                /*       FILE = 'VOCAB';                                               */ THP0220
                /*       FILE = 'DRCTRY';                                              */ THP0230
                /*       FILE = 'THES', INTERPRET = 'YES';                             */ THP0240
                /*       FILE = 'CONTROL';                                             */
                /*       FILE = 'V';  FILE = 'T', INTERPRET = 'Y';                     */
                /* THE DEFAULT OPTION FOR INTERPRET IS 'NO'.  INTERPRET = 'YES'        */ THP0250
                /* CAUSES THE DRCTRY & VOCAB ENTRIES TO BE FETCHED AND PRINTED FOR     */ THP0260
                /* EACH THES ENTRY.  RUN TIME IS CONSIDERABLY LONGER.                  */ THP0270
                /***********************************************************************/ THP0280
 2                     DCL VOCAB  FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,    THP1000
                           DRCTRY FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,    THP1010
                           THES   FILE RECORD ENVIRONMENT(REGIONAL(1)) KEYED DIRECT,    THP1020
                           THSCTL FILE STREAM INPUT;                                    THP1030
 3                     DCL  VBLKSIZE   FIXED BIN,     /* # RECORDS PER BLOCK         */ THP1040
                            DBLKSIZE   FIXED BIN,                                       THP1050
                            TBLKSIZE   FIXED BIN,                                       THP1060
                            VLASTBLK   FIXED BIN,     /* # OF LAST BLOCK IN VCC -    */ THP1070
                                                      /*   THE OVERFOLW BUCKET.      */ THP1080
                            DLASTBLK   FIXED BIN,     /* # OF LAST BLOCK IN DIR      */ THP1090
                            TLASTBLK   FIXED BIN;     /* # OF LAST BLOCK IN THS      */ THP1100
 4                     DCL  VKEE       FIXED DEC(5),  /* VOCABULARY BLOCK KEY        */ THP1110
                            DKEY       FIXED DEC(5),  /* DIRECTORY  BLOCK KEY        */ THP1120
                            TKEY       FIXED DEC(5);  /* THESAURUS  BLOCK KEY        */ THP1130
 5                     DCL  VOC@       FIXED BIN INITIAL(-1),   /* DATA SET INDICES.*/ THP1134
                            DIR@       FIXED BIN INITIAL(-1),                           THP1135
                            THS@       FIXED BIN INITIAL(-1);                           THP1135
 6                     DCL  VINCOREKEY FIXED DEC(5) INITIAL(-1);                        THP1136
 7                     DCL  01 VOCBLOCK CONTROLLED,   /* VOCABULARY ENTRY            */ THP1140
                              02 VOCRCD(0:VBLKSIZE-1),                                  THP1150
                                 03 VM        FIXED BIN,                                THP1190
                                 03 VS        FIXED BIN,                                THP1200
                                 03 VDIR@     FIXED BIN,                                THP1210
                                 03 VC        FIXED BIN,                                THP1220
                                 03 VX        CHAR(1),                                  THP1225
                                 03 VF        CHAR(1),                                  THP1230
                                 03 VWORD     CHAR(18);                                 THP1240
 8                     DCL  01 THSBLOCK CONTROLLED,   /* THESAURUS DATA SET.         */ THP1250
                              02 THSRCD(0:TBLKSIZE-1),/*   WE DO OUR OWN BLOCKING.   */ THP1260
                                 03 TDIR@     FIXED BIN,                                THP1270
                                 03 TVOC@     FIXED BIN;                                THP1280
 9                     DCL  01 DIRBLOCK CONTROLLED,   /* DIRECTORY DATA SET.         */ THP1290
                              02 DIRRCD(0:DBLKSIZE-1),/*   WE DO OUR OWN BLOCKING.   */ THP1300
```

```
                                  03 DCAT      CHAR(8),                                        THP1310
                                  03 DTHS@     FIXED BIN,                                      THP1320
                                  03 DLNG      FIXED BIN;                                      THP1330
 10                    DCL  VKEY(26,26) FIXED DEC(4),        /* VOCAB BUCKET KEYS        */ THP1340
                            VEXT(26,26) FIXED DEC(4);        /*   AND EXTENTS.           */ THP1350
 11                    DCL  (FILE, INTERPRET)  CHAR(1);      /* CONTROL CARD PARAMS.     */ THP1360
 12           BEGIN:   GET FILE(THSCTL)DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE,                        THP3000
                                           VLASTBLK,DLASTBLK,TLASTBLK);                        THP3010
 13                    PUT SKIP(2); PUT DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE,                       THP3012
                                        VLASTBLK,DLASTBLK,TLASTBLK);                           THP3013
 15                    GET FILE(THSCTL)EDIT(VKEY,VEXT)(F(4));                                  THP3014
 16                    ALLOCATE VOCBLOCK, THSBLOCK, DIRBLOCK;                                  THP3020
 17                    ON ENDFILE(SYSIN)  GO TO ENDTHSPRNT;                                    THP3021
 19           CONTROL_CARD: FILE = ' '; INTERPRET = ' ';                                       THP3022
 21                    GET DATA;                                                               THP3023
 22                    IF FILE = 'C' THEN GO TO PCON;
 24                    IF FILE = 'V' THEN GO TO PVOC;                                          THP3024
 26                    IF FILE = 'D' THEN GO TO PDIR;                                          THP3025
 28                    IF FILE = 'T' THEN GO TO PTHS;                                          THP3026
 30                    GO TO CONTROL_CARD;                                                     THP3027
 31           PCON:    PUT EDIT('*** CONTROL DATA ***')(PAGE,A);
 32                    PUT SKIP(2); PUT DATA(VBLKSIZE,DBLKSIZE,TBLKSIZE,
                                        VLASTBLK,DLASTBLK,TLASTBLK);
 34                    PUT SKIP; PUT DATA(VKEY); PUT SKIP; PUT DATA(VEXT);                     THP3015
 38                    GO TO CONTROL_CARD;
 39           PVOC:    PUT EDIT('*** VOCABULARY ***')(PAGE,A);                                 THP3030
 40                    PUT EDIT('  VOC@ IVOC       MATCNT       SECT',                         THP3034
                            '        DIR@      COUNT FLAG  TYPE')(SKIP(2),A,A);                THP3035
 41                    DO VKEE = 0 TO VLASTBLK;                                                THP3040
 42                      READ FILE(VOCAB)INTO(VOCBLOCK)KEY(VKEE);                              THP3050
 43                      PUT EDIT('** VOC BLOCK ',VKEE)(SKIP(2),A,F(7));                       THP3060
 44                      DO I = 0 TO VBLKSIZE-1;                                               THP3070
 45                        VOC@ = VOC@ + 1;                                                    THP3075
 46                        IF VWORD(I) = ' ' THEN GO TO VSCANX;                                THP3079
 48                        PUT EDIT(VOC@,'.',I,'.',VM(I),',',VS(I),',',                        THP3080
                                     VDIR@(I),',',VC(I),',',VF(I),',',VWORD(I))                THP3090
                                    (SKIP,F(6),A,F(4),A,X(2),4(F(9),A),X(3),A(1),A,            THP3100
                                     A(18));                                                   THP3110
 49           VSCANX: END; END;                                                                THP3120
 51                    GO TO CONTROL_CARD;                                                     THP3121
 52           PDIR:    PUT EDIT('*** DIRECTORY ***')(PAGE,A);                                  THP3130
 53                    PUT EDIT('  DIR@ IDIR   CATEGORY       THS@     LENGTH')                 THP3134
                               (SKIP(2),A);                                                    THP3135
 54                    DO DKEY = 0 TO DLASTBLK;                                                THP3140
 55                      READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEY);                             THP3145
 56                      PUT EDIT('** DIR BLOCK ',DKEY)(SKIP(2),A,F(7));                       THP3150
 57                      DO I = 0 TO DBLKSIZE-1;                                               THP3160
 58                        DIR@ = DIR@ + 1;                                                    THP3162
 59                        IF DCAT(I) = '99999999' THEN GO TO DSCANX;                          THP3169
 61                        PUT EDIT(DIR@,'.',I,'.',DCAT(I),',',DTHS@(I),',',DLNG(I))           THP3170
                                    (SKIP,F(6),A,F(4),A,X(2),A(8),A,F(9),A,F(9));              THP3180
 62           DSCANX: END; END;                                                                THP3190
 64                    GO TO CONTROL_CARD;                                                     THP3191
```

```
PTHS:   PUT EDIT('*** THESAURUS ***')(PAGE,A);                                  THP3200
        PUT EDIT('  THS@ ITHS          DIR@       VOC@')(SKIP(2),A);            THP3205
        IF INTERPRET = 'Y' THEN                                                 THP3206
          DO; PUT EDIT('CATEGCRY','WORD')(COLUMN(44),A,COLUMN(62),A);           THP3207
              IDIR = 0; DKEY = 0; L = 0;                                        THP3208
              READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEY);                         THP3209
          END;                                                                  THP3210
        DO TKEY = 0 TO TLASTBLK;                                                THP3213
          READ FILE(THES)INTO(THSBLOCK)KEY(TKEY);                               THP3215
          PUT EDIT('** THS BLCCK ',TKEY)(SKIP(2),A,F(7));                       THP3220
          DC I = 0 TO TBLKSIZE-1;                                               THP3230
            THS@ = THS@ + 1;                                                    THP3232
            IF TVOC@(I) = -1 THEN GO TO TSCANX;                                 THP3239
            PUT EDIT(THS@,'.',I,'.',TDIR@(I),',',TVCC@(I))                      THP3240
                    (SKIP,F(6),A,F(4),A,X(2),F(9),A,F(9));                      THP3250
            IF INTERPRET = 'Y' THEN                                             THP3260
              DO; L = L + 1;                                                    THP3270
                  IF L > DING(IDIR) THEN                                        THP3272
                    DO; IDIR = IDIR + 1;                                        THP3274
                        IF IDIR = DBLKSIZE THEN                                 THP3276
                          DO; DKEY = DKEY + 1;                                  THP3278
                              READ FILE(DRCTRY)INTO(DIRBLOCK)KEY(DKEY);         THP3280
                              IDIR = 0;                                         THP3282
                          END;                                                  THP3284
                        L = 1;                                                  THP3286
                    END;                                                        THP3288
                  VKEE = TVOC@(I) / VBLKSIZE;                                   THP3290
                  IVOC = TVOC@(I) - (VKEE*VBLKSIZE);                            THP3300
                  IF VINCOREKEY ¬= VKEE THEN                                    THP3350
                    DO; READ FILE(VOCAB) INTO(VCCBLCCK) KEY(VKEE);              THP3360
                        VINCOREKEY = VKEE;                                      THP3370
                    END;                                                        THP3380
                  IF L=1 THEN PUT EDIT(DCAT(IDIR))(COLUMN(44),A(8));            THP3390
                  PUT EDIT(VWCRD(IVOC))(CCLUMN(62),A(58));                      THP3400
              END;                                                              THP3410
TSCANX: END; END;                                                               THP3420
        GO TO CONTROL_CARD;                                                     THP3421
ENDTHSPRNT: END THSPRNT;                                                        THP3430
```

```
VBKTCNT: PRCC OPTIONS(MAIN);                                                                          PAGE    2

   1             VBKTCNT: PROC CPTICNS(MAIN);
                 /**********************************************************************/
                 /* PRODUCES DATA TO AID IN DETERMINING VOCAB BUCKET SIZES AND BLOCK- */
                 /* ING SIZE.                                                          */
                 /* INPUT: "ORGNX".  RING-STRUCTURE THESAURUS LIST IN INDEX ORDER.     */
                 /*         CARD IMAGES.  FORMAT:                                      */
                 /*            CC 01-08: CATEGCRY DESIGNATIONS                         */
                 /*            CC 09:    COMMA                                         */
                 /*            CC 10-80: WORD(TOKEN)                                   */
                 /* PRINTED OUTPUT:  LIST OF BUCKETS THAT WILL APPEAF IN VOCAB. AND # */
                 /*                  CF ENTRIES IN EACH.                               */
                 /**********************************************************************/
   2                     DCL ORGNX FILE INPUT STREAM;
   3                     DCL PAIR      CHAR(2),
                             WORD      CHAR(71),
                             WORDSAV   CHAR(71);
   4                     PUT EDIT('VOCAB BUCKETS')(PAGE,A);  PUT SKIP;
   6                     ON ENDFILE(ORGNX) GO TO LASTBKT;
   8                     GET FILE(ORGNX)EDIT(WCRD) (X(9),A(71));
   9                     ITOKENS = 1; ITYPES = 1
  11                     WORDSAV = WORD;
  12                     IBKTCNT = 0;  IBKTSIZE = 1
  14                     PAIR = SUBSTR(WORD,1,2);
  15             GETN:   GET FILE(ORGNX)EDIT(WCRD)(X(09),A(71));
  16                     ITOKENS = ITCKENS + 1;
  17                     IF PAIR ¬= SUBSTR(WORD,1,2) THEN
  18                       DO; PUT EDIT(PAIR,IBKTSIZE)(X(12),A(2),F(6));
  20                           IBKTSIZE = 0;
  21                           IBKTCNT = IBKTCNT + 1;
  22                           PAIR = SUBSTR(WCRD,1,2);
  23                       END;
  24                     IF WORDSAV ¬= WORD THEN
  25                       DO; ITYPES = ITYPES + 1;
  27                           IBKTSIZE = IBKTSIZE + 1;
  28                           WORDSAV = WORD;
  29                       END;
  30                     GO TO GETN;
  31             LASTBKT:  PUT EDIT(PAIR,IBKTSIZE)(X(12),A(2),F(6));
  32                     IBKTCNT = IBKTCNT + 1;
  33                     PUT EDIT(IBKTCNT,' BUCKETS')(SKIP(2),F(7),A);
  34                     PUT EDIT(ITYPES,' TYPES')(SKIP(2),F(7),A);
  35                     PUT EDIT(ITOKENS,' TOKENS')(SKIP(2),F(7),A);
  36             END;
  1                CPTBS:  EROC OPTIONS(MAIN);
                   /*********************************************************************/
                   /* THIS FROCEDURE IS AN AID IN COMPUTING OPTIMUM EICCK SIZES FOR THE */
                   /* "THES" DATA SET IN THE RING-STRUCTURE THESAURUS.                  */
                   /*                                                                   */
                   /* THE PROBLEM IS THAT A SELECTION OF TBLKSIZE HAS TWO CONSEQUENCES: */
                   /* (1) THE NUMEER OF BLOCKS IN THE    (2) THE NUMBER OF CATEGORIES   */
                   /* THAT OVERLAP INTO MORE THAN ONE BLOCK.  IT IS DESIRABLE TO        */
                   /* MINIMIZE DISK ACCESSES, AND THUS TO MINIMIZE THE SUM OF (# BLOCKS */
                   /* + # OVERLAPS).  WE WILL CALL THIS SUM "THE CONSEQUENCES" CF A     */
                   /* TBLKSIZE CHOICE.  THIS PROGRAM COMPUTES THE CONSEQUENCES OF EACH  */
                   /* POSSIBLE TBLKSIZE.  THEN, EASED ON THE "OPT" OPTION, EITHER LISTS */
                   /* THEM ALL OR PICKS THE 25 LOWEST CONSEQUENCES AND PRINTS THEM WITH */
                   /* THEIR TBLKSIZES. FROM THIS THE USER MAY MAKE HIS FINAL CHOICE,    */
                   /* POSSIELY AUGMENTED BY ADDITIONAL INFORMATION ON DISK CAPACITY, ETC*/
                   /*                                                                   */
                   /* INPUT  - SYSIN: IN DATA FORMAT: "N", "MAXTBLKSIZE", & "OPT";      */
                   /*                    "N" IS THE DIMENSION OF THE VECTOR "X".        */
                   /*                    "MAXTBLKSIZE" IS THE MAXIMUM ALLOWABLE TBLKSIZE*/
                   /*                    "OPT" DESIGNATES CHOICE OPTION.  VALUES ARE    */
                   /*                          "LIST" AND "SELECT".  "LIST" CAUSES ALL  */
                   /*                          RESULTS TO BE LISTED; "SELECT" CAUSES    */
                   /*                          THE PROGRAM TO SELECT THE BEST 25 CHOICES*/
                   /*                          OF TBLKSIZE AND TO LIST THEM, TOGETHER   */
                   /*                          WITH THEIR CONSEQUENCES.                 */
                   /*                 IN LIST FORMAT: THE VECTOR, "X", WHOSE ENTRIES ARE*/
                   /*                          THE LENGTHS, IN # OF ENTRIES, OF THE     */
                   /*                          CATEGORIES IN THES, IN THE ORDER IN WHICH*/
                   /*                          THEY APPEAR IN THE THES DATA SET!        */
                   /* OUTPUT - A LISTING OF THE TBLKSIZES (IN # RCDS & # BYTES) AND     */
                   /*          THEIR CONSEQUENCES (# OF BLOCKS AND # OF OVERLAPS).  IF  */
                   /*          THE INPUT OPTION "LIST" IS SPECIFIED, ALL RESULTS WILL   */
                   /*          BE PRINTED; IF "SELECT" IS SPECIFIED, THE "BEST" 25 (OR  */
                   /*          LESS) WILL BE PRINTED.                                   */
                   /*                                                                   */
                   /* SAMPLE INPUT:                                                     */
                   /* N=5, MAXTBLKSIZE = 7200, OPT ='L';                                */
                   /* 201 150 330 450 20                                                */
                   /*********************************************************************/
  2                     DCL ( X(N),                        /* VECTOR OF THES CAT LENGTHS  */
                              BLOCKS(MAXTBLKSIZE),         /* # BLOCKS FOR EACH TBLKSIZE  */
                              OVLAPS(MAXTBLKSIZE)          /* # OVLAPS FOR EACH TBLKSIZE  */
                            )  FIXED BINARY CONTROLLED;
  3                     DCL ( MAXTBLKSIZE,                 /* ENTERED IN BYTES AS DATA    */
                              N,                           /* ENTERED AS DATA - DIMENSION */
                                                           /* OF "X".                     */
                              SM,                          /* WORKING BLOCKSIZE           */
                              WT(26), NX(26), WTX, NXX  /* WTS & INDEXES FOR SELEC-    */
                                                           /* TION.                       */
                            )  FIXED BIN;
  4                     DCL   CPT   CHAR(1) INITIAL('L');  /* ENTERED AS DATA -         */
                                                             /* CHOICE OPTION.            */
  5                     PUT PAGE;
```

```
            ON ENDFILE (SYSIN) GO TO ENDOPTBS;
    GET:    GET DATA;                          /* READ N, MAXTBLKSIZE, OPT    */
            MAXTBLKSIZE = MAXTBLKSIZE/8;  /* CONVERT FROM BYTES TO RCDS  */
            ALLOCATE X, BLOCKS, OVLAPS;
            GET LIST (X);                      /* READ VECTOR OF CAT LENGTHS  */
            OVLAPS = 0;
      /* COMPUTE THE CONSEQUENCES FOR EACH TBLKSIZE.                          */
            DO L=MAXTBLKSIZE TO 2 BY -1;
              SM=0; BLOCKS(L)=1; I=1;
    CONTINUE:
              DO I=I TO N WHILE(SM<L);
                SM = SM + X(I);
              END;
              IF (I=N+1)&(SM<L) THEN GO TO NEXTL;
              SM = SM - L;
              IF SM ¬= 0 THEN OVLAPS(L) = OVLAPS(L) + 1;
              BLOCKS(L) = BLOCKS(I) + 1;
              GO TO CONTINUE;
    NEXTL:  END;
      /* HERE WE HAVE COMPUTED ALL THE CONSEQUENCES.  "OPT" TELLS US WHAT*/
      /* TO DO WITH THEM.                                                     */
            IF OPT = 'S' THEN GO TO SELECT;
    LIST:   /* LIST ALL THE CONSEQUENCES.                                     */
            PUT EDIT('**** TBLKSIZE BLOCKS OVLAPS')(SKIP(2),A);
            PUT EDIT((L,L*8,BLOCKS(L),OVLAPS(L)
              DO L = MAXTBLKSIZE TO 2 BY -1))
              (SKIP,F(4),F(9),F(7),F(7));
            FREE X, BLOCKS, OVLAPS;
            GO TO GET;
    SELECT: /* SELECT 25 TBLKSIZES WITH "BEST" CONSEQUENCES AND LIST.    */
            IMAX=0; NX=0; WT=9999;            /* PICK TOP 25.                 */
            DO L=MAXTBLKSIZE TO 2 BY -1;
              SM = BLOCKS(L) + OVLAPS(L);
              INSERT = 0;
              DO I = IMAX TO 1 BY -1 WHILE(SM<WT(I));
                WT(I+1) = WT(I); NX(I+1) = NX(I);
                INSERT = 1;
              END;
              IF INSERT=1 THEN DO; WT(I+1)=SM; NX(I+1)=I; END;
              IF IMAX < 25 THEN
                DO; IMAX = IMAX+1;
                    IF INSERT=0 THEN DO; WT(IMAX)=SM; NX(IMAX)=L; END;
                END;
            END;
            PUT EDIT('**** ',IMAX,' "BEST" TBLKSIZE CHOICES:') /* PRINT  */
                    (SKIP(2),A,F(3),A);                /* TOP 25 (OR LESS) */
            PUT EDIT('      TBLKSIZE BLOCKS OVLAPS')(SKIP,A);
            PUT EDIT((NX(I),NX(I)*8,BLOCKS(NX(I)),OVLAPS(NX(I))
                     DO I=1 TO IMAX))
                    (SKIP,F(4),F(9),F(7),F(7));
            FREE X, BLOCKS, OVLAPS;
            GO TO GET;
    ENDOPTBS: END OPTBS;
```

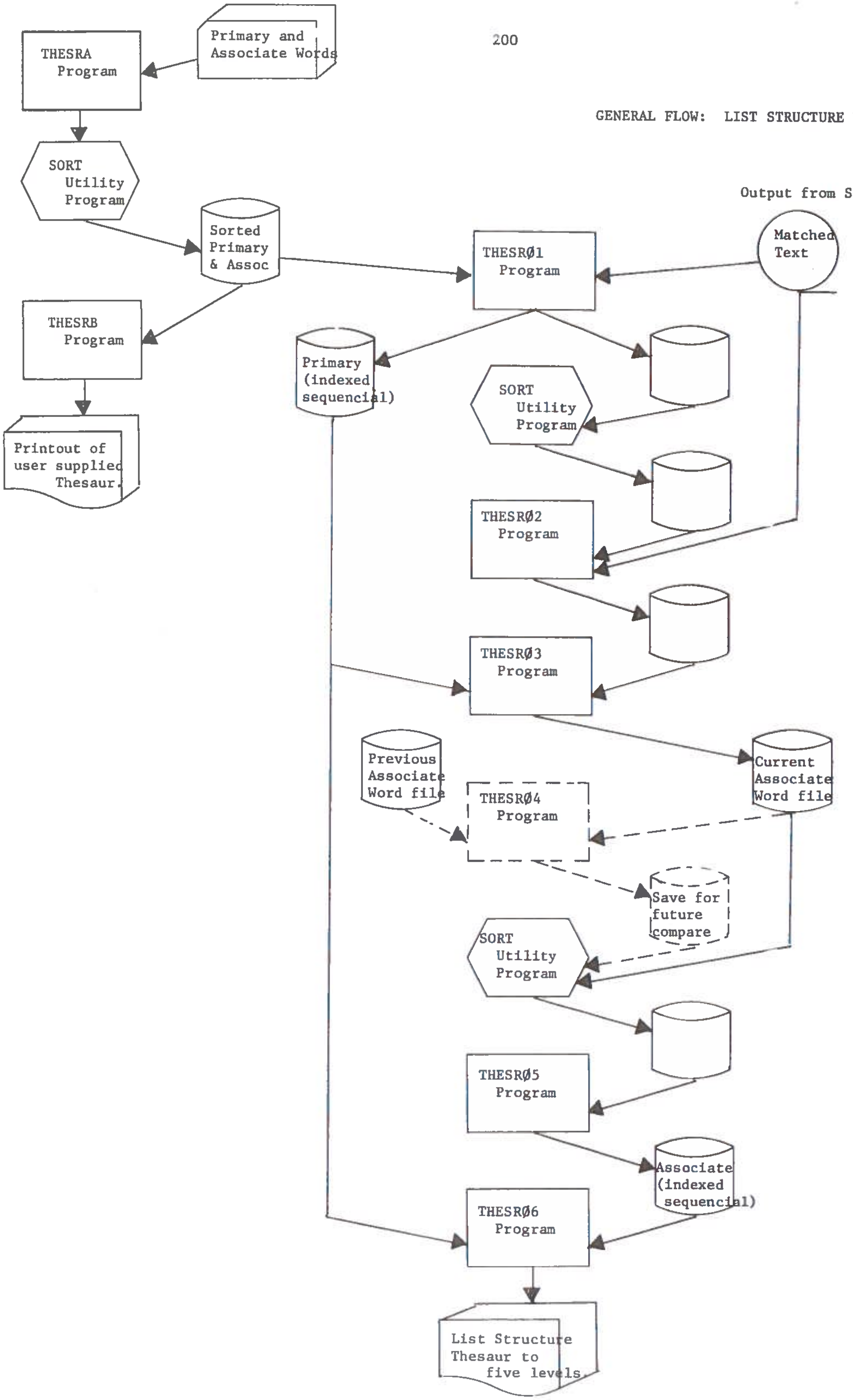
APPENDIX D

List-Structure VIA Programs

by

William G. Hickok

GENERAL FLOW: LIST STRUCTURE VIA



Compare Primary words
to Content words from
text.

Sort Associate words
into ascending sequence.

Compare Associate words
to Content words from
text.

Compare Associate words
to Primary words.

USER OPTION:
Compare current Associate
word data set to
previously processed
Associate word file.

Sort on the Back link
to the Primary words.

Write the Associate
word file as an
Indexed Sequencial
file.

Print the Primary and
Associate words which
appear in text showing
the relationship
between Primary and
Associate to five levels.

```
STMT LEVEL NEST
                    /* THESRA:  PROCEDURE TO CREATE ORIGINAL THESAURS.                       */

                    /************************************************************************/
                    /*                                                                      */
                    /*  THE THESRA PROGRAM PRODUCES A THESAURS FILE FROM USER SUPPLIED      */
                    /*  INPUT.  THE DATA SET CONSISTS OF WORD PAIRS WITH EACH WORD PAIF     */
                    /*  CONSISTING OF A PRIMARY WORD FOLLOWED BY AN ASSOCIATED WORD.        */
                    /*                                                                      */
                    /*                                                                      */
                    /*  INPUT CARD FORMAT:                                                  */
                    /*                                                                      */
                    /*    INPUT IS FREE FORMAT WITH A PRIMARY WORD BEGINNING IN CARD        */
                    /*      COLUMN ONE FOLLOWED BY AN ASSOCIATED WORD ONE BLANK AFTER       */
                    /*      THE LAST CHARACTER OF THE PRIMARY WORD.  EACH WORD PAIR         */
                    /*      BEGINS A NEW CARD.                                              */
                    /*                                                                      */
                    /*                                                                      */
                    /*  OUTPUT FORMAT:                                                      */
                    /*                                                                      */
                    /*    OUTPUT IS TO A TEMPERARY DATA SET WHICH, IN TURN, IS              */
                    /*      INTRODUCED TO SORT.  RECORD FORMAT FOLLOWS:                     */
                    /*                                                                      */
                    /*      POSITION  FIELD CONTENTS                                        */
                    /*        01-02   PRIMARY WORD LENGTH                                   */
                    /*        03-20   PRIMARY WORD                                          */
                    /*        21-22   ASSOCIATE WORD LENGTH                                 */
                    /*        23-40   ASSOCIATE WORD                                        */
                    /*                                                                      */
                    /*                                                                      */
                    /*  SUGGESTED JCL:                                                      */
                    /*                                                                      */
                    /*    1              16                                                 */
                    /*      //GO.OUTPUT DD DSNAME=&THESTEMP,UNIT=HDSKO,DISP=(NEW,PASS),     */
                    /*      //           DCB=(RECFM=FB,LRECL=40,BLKSIZE=7200),              */
                    /*      //           SPACE=(TRK,(50,10))                                */
                    /*      //GO.SYSIN  DD *                                                */
                    /*        USER SUPPLIED CARD INPUT                                      */
                    /*      /*                                                              */
                    /*                                                                      */
                    /*  SORTING OF OUTPUT:                                                  */
                    /*                                                                      */
                    /*    THE FOLLOWING IS THE JCL AND SORT CONTROL CARDS TO SORT           */
                    /*      THE OUTPUT FROM THESRA.                                         */
                    /*                                                                      */
                    /*      //STEP2  EXEC  PGM=IERRC000                                     */
                    /*      //SYSOUT     DD SYSOUT=A                                        */
                    /*      //SYSPRINT   DD SYSOUT=A                                        */
                    /*      //SORTLIB    DD DSNAME=SYSL.SORTLIB,DISP=SHR,VOLUME=REF=PACK1,  */
                    /*      //SORTIN     DD DSNAME=&THESTEMP,DISP=(OLD,DELETE)              */
                    /*      //SORTOUT    DD DSNAME=UNC.IS.P2312.SEDELOW.ORGTHES,            */
                    /*      //              DISP=(OLD,KEEP),UNIT=2314,VOLUME=SER=USWLIB     */
                    /*      //SORTWK01   DD UNIT=HDSKO,SPACE=(TRK,(50),,CONTIG)             */
```

```
STMT LEVEL NEST

                  /*      //SORTWK02  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)              */
                  /*      //SORTWK03  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)              */
                  /*      //SORTWK04  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)              */
                  /*      //SORTWK05  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)              */
                  /*      //SORTWK06  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)              */
                  /*      //SYSIN     DD *                                                */
                  /*       SORT FIELDS=(3,18,CH,A,23,18,CH,A),SIZE=E20000                 */
                  /*      /*                                                              */
                  /*                                                                      */
                  /*  SORT SHOULD BE FOLLOWED BY THESRB WHICH PRINTS THE THESAUR          */
                  /*  FOR FUTURE HARI COPY REFERENCE.                                     */
                  /*                                                                      */
                  /**********************************************************************/

   1              THESRA:   PROCEDURE OPTIONS (MAIN);

   2      1             DECLARE
                          CARDIMAGE (80) CHAR (1) STATIC,   /* CARD READ IN AREA          */
                          AWORD CHAR (18) VARYING,       /* WORD FORMATION AREA           */
                          COL    FIXED DEC (2) INIT (73),/* COLUMN COUNTER                */
                          NUMC   FIXEL DEC (2);          /* WORD LENGTH                   */

   3      1             OPEN FILE(SYSIN) INPUT,
                            FILE(OUTPUT) OUTPUT;
   4      1             ON ENDFILE(SYSIN) GO TO FINI;

   6      1       RPW:  COL = 73;
   7      1             CALL FORM;
   8      1             PUT FILE(OUTPUT) EDIT (NUMC,AWCRD)   (F(2),A(18));  /* PRIMARY  */
   9      1             CALL FORM;
  10      1             PUT FILE(OUTPUT) EDIT (NUMC,AWURD)  (F(2),A(18));  /* ASSOCIATED*/
  11      1             GO TO RPW;

  12      1       FINI: CLOSE FILE(OUTPUT), FILE(SYSIN);


                  /**********************************************************************/
                  /*                                                                      */
                  /* SUBROUTINE 'FORM'.  SUBROUTINE OBTAINS PRIMARY AND ASSCOIATED        */
                  /*                     WORDS FROM AN INPUT RECORD USING A BLANK AS A    */
                  /*                     WORD DELIMITER.                                  */
                  /*                                                                      */
                  /**********************************************************************/

  13      1       FORM:   PRCCEDURE;
  14      2             NUMC = 0;
  15      2             AWORD = '';

  16      2       BUMP: COL = COL + 1;
  17      2             IF COL <= 72 THEN GO TO EXTRACT;
```

```
      /* THESRA:  PROCEDURE TO CREATE ORIGINAL THESAURS.             */                    PAGE     4

STMT LEVEL NEST

 19     2          READ: GET FILE(SYSIN) EDIT (CARDIMAGE) (80 A(1));
 20     2                COL = 1;

 21     2          EXTRACT:
 22     2                IF CARDIMAGE(COL) = ' ' THEN GO TO LSTCHAR;
 23     2                NUMC = NUMC + 1;
 24     2                AWORD = AWORD || CARDIMAGE(COL);
 25     2                GO TO BUMP;

 26     2          LSTCHAR:
 27     2                IF NUMC = 0 THEN GO TO BUMP;

 28     2                RETURN;

 29     2                END FORM;

                   /************************************************************************/

 30     1                END THESRA;
```

```
STMT LEVEL NEST
                  /*  THESRB:  PROCEDURE TO PRINT OUTPUT OF THESRA.                     */

                  /**********************************************************************/
                  /*                                                                    */
                  /*  THESRB PROGRAM PRINTS THE OUTPUT OF THE THESRA PROGRAM AFTER      */
                  /*  SORT.  EXECUTICN OF THIS PROGRAM IS A USER OPTION AND IS NOT      */
                  /*  NECESSARY FOR THE SUCCESSFUL COMPLETION OF THE THESAUR PROGRAM    */
                  /*  PACKAGE.                                                          */
                  /*                                                                    */
                  /*  INPUT MUST HAVE BEEN SORTED BEFORE INTRODUCTION TO THIS PROGRAM.  */
                  /*  THE USER IS REFERRED TO THE THESRA PROGRAM DOCUMENTATION FOR      */
                  /*  INPUT RECORD FCRMAT AND SORT DOCUMENTATION.                       */
                  /*                                                                    */
                  /**********************************************************************/

  1               THESRB:  PROCEDURE OPTIONS(MAIN);

  2     1               DECLARE
                          SAVEPRIM CHAR (18) VARYING INITIAL (' '),
                          PRIMWD   CHAR (18),
                          ASSOCWD  CHAR (18),
                          SEQ1     FIXED DEC (4) INITIAL (0),
                          SEQ2     FIXED DEC (4) INITIAL (0),
                          TOTAL    FIXED DEC (5) INITIAL (0),
                          X        FIXED BIN (15,0) INITIAL (0),
                          COUNTER  FIXED DEC (4) INITIAL (0);

  3     1               OPEN FILE(SYSPRINT) OUTPUT, FILE(INPUT) INPUT;
  4     1               ON ENDFILE(INPUT) GO TO FINISH;
                                                        /********************************/
  6     1               ON ENDPAGE(SYSPRINT) BEGIN;     /*                              */
  8     2                 CALL PGHDG;                   /*  SET UP END OF PAGE          */
  9     2                 END;                          /*  CONDITION.                  */
                                                        /*                              */
                                                        /********************************/

 10     1               CALL PGHDG;   /* FIRST PAGE INITIALIZATION                      */

 11     1         RDWD:
                        GET FILE(INPUT) EDIT (PRIMWD,ASSOCWD) (X(2),A(18),X(2),A(18));
 12     1         PRTASSOC:
 13     1               IF SAVEPRIM = PRIMWD THEN DO;
 14     1                 SEQ2 = SEQ2 + 1;
 15     1                 PUT EDIT (SEQ2,ASSOCWD) (SKIP(X),COLUMN(26),F(4),X(2),
                                A(18));
 16     1                 X = 1;
 17     1                 GO TO RDWD;
 18     1                 END;

 19     1               IF SAVEPRIM ¬= PRIMWD THEN DO;
```

```
          /*  THESRB:  PROCEDURE TC PRINT OUTPUT OF THESPA.                    */

STMT LEVEL NEST

 21     1                SEQ1 = SEQ1 + 1;
 22     1                PUT EDIT (SEQ1,PRIMWD)  (SKIP(2),F(4),X(2),A(18));
 23     1                X = 0;
 24     1                TOTAL = TOTAL + SEQ2;
 25     1                SFQ2 = 0;
 26     1                SAVEPRIM = PRIMWD;
 27     1                END;

 28     1              GO TO PRTASSOC;

 29     1          FINISH:
                       PUT EDIT ('TOTAL PRIMARY WORDS = ',SEQ1,
                              ';  TOTAL ASSOCIATE WORDS = ',TOTAL)
                              (SKIP(3),A,F(4),A,F(5));


                   /**************************************************************************/
                   /*                                                                        */
                   /*  PAGE HEADING PROCEDURE TO PRINT HEADING AT TOP OF EACH OUTPUT         */
                   /*  PAGE.                                                                 */
                   /*                                                                        */
                   /**************************************************************************/

 30     1          PGHDG:  PROCEDURE;

 31     2                COUNTER = COUNTER + 1;
 32     2                PUT EDIT ('COUNT  PRIMARY WCRD','COUNT  ASSOCIATED WORD',
                               'PAGE',COUNTER)  (PAGE,A,COLUMN(26),A,COLUMN(50),A,F(4));

 33     2                PUT FILE(SYSPRINT) SKIP(1);
 34     2                END PGHDG;

                   /**************************************************************************/
```

```
STMT LEVEL NEST
                /* THESR01:  PROCEDURE TO COMPARE CONTENT WORDS TO  PRIMARY WORDS.   */

                /*********************************************************************/
                /*                                                                   */
                /*  GENERAL PROGRAM FLOW:                                            */
                /*                                                                   */
                /*  THE THESR01 PROGRAM COMPARES THE DATA SET OF PRIMARY AND         */
                /*  ASSOCIATED WORDS PRODUCED BY THESRA TO THE DATA SET OF CONTENT   */
                /*  WORDS PRODUCED BY THE PROGRAM SUFFIX.                            */
                /*                                                                   */
                /*    COMPARE PRIMARY WORD WITH OUTPUT FROM THE SUFFIX PROGRAM.  IF  */
                /*    WORDS MATCH THEN OUTPUT PRIMARY WORD TO (OUTPUT1) USING THE    */
                /*    INDEXED SEQUENTIAL DATA SET ORGANIZATION, AND OUTPUT THOSE     */
                /*    ASSOCIATED ASSOCIATE WORDS TO (OUTPUT2).  OUTPUT OF ASSOCIATE  */
                /*    WORDS OCCURS ONLY WHEN THERE IS A MATCH OF THE PRIMARY WORD    */
                /*    TO A CONTENT WORD.  WHEN A DIFFERENT FORM OF THE PRIMARY       */
                /*  WORD APPEARS IN THE TEXT, THE FIELD 'PRMSTAT' IS SET EQUAL TO    */
                /*  THE VALUE ONE.                                                   */
                /*                                                                   */
                /*                                                                   */
                /*  INPUT:                                                           */
                /*                                                                   */
                /*    INPUT FORMAT FOR THE USER SUPPLIED THESAUR, OUTPUT FROM THE    */
                /*    THESRA PROGRAM (INPUT1);                                       */
                /*                                                                   */
                /*      POSITION  FIELD DESCRIPTION                                  */
                /*        01-02   PRIMARY WORD LENGTH                                */
                /*        03-20   PRIMARY WORD                                       */
                /*        21-22   ASSOCIATE WORD LENGTH                              */
                /*        23-40   ASSOCIATED WORD                                    */
                /*                                                                   */
                /*    THE ABOVE FILE MUST HAVE BEEN SORTED ON PRIMARY WORD PRIOR TO  */
                /*    INTRODUCTION TO THIS PROGRAM.  THE SORT CONTROL CARD IS:       */
                /*      SORT FIELDS=(3,18,CH,A,23,18,CH,A),SIZE=E20000               */
                /*                                                                   */
                /*    INPUT FORMAT OF THE CONTENT WORD DATA SET, OUTPUT FROM THE     */
                /*    SUFFIX PROGRAM (INPUT2);                                       */
                /*                                                                   */
                /*      POSITION  FIELD DESCRIPTION                                  */
                /*        01-02   CONTENT WORD LENGTH                                */
                /*        03-07   MATCH COUNT LINKING WORDS OF COMMON ROOT           */
                /*        08-12   FREQUENCY OF OCCURRENCE OF WORD                    */
                /*        13-30   CONTENT WORD                                       */
                /*                                                                   */
                /*    THE ABOVE OUTPUT MUST HAVE BEEN SORTED ON MATCH COUNT AND      */
                /*    WITHIN MATCH COUNT ON WORD PRIOR TO INTRODUCTION TO THIS       */
                /*    PROGRAM:                                                       */
                /*      SORT FIELDS=(3,5,CH,A,13,18,CH,A),SIZE=E20000                */
                /*                                                                   */
                /*                                                                   */
                /*  OUTPUT:                                                          */
                /*                                                                   */
```

```
STMT LEVEL NEST

                  /*  OUTPUT OF THESR01 IS TWO DATA SETS.  THE FIRST, (OUTPUT1),         */
                  /*  CONTAINS ALL PRIMARY WORDS WHICH ARE ALSO CONTENT WORDS.  THE      */
                  /*  SECOND, (OUTPUT2) CONTAINS ALL ASSOCIATE WORDS WHICH ARE           */
                  /*  ASSOCIATED WITH THE PRIMARY WORDS WRITTEN IN (OUTPUT1)             */
                  /*                                                                     */
                  /*    FORMAT OF THE OUTPUT1 DATA SET;                                  */
                  /*                                                                     */
                  /*      POSITION  FIELD DESCRIPTION                                    */
                  /*        01-02   LENGTH OF PRIMARY WORD                               */
                  /*           03   STATUS OF WORD  EQUALS 1 IF APPEARS IN DIFFERENT     */
                  /*                FORM IN TEXT.  0 OTHERWISE.                          */
                  /*        04-07   FREQUENCY OF OCCURRENCE                              */
                  /*        08-11   MATCH COUNT                                          */
                  /*        12-15   FORWARD LINK TO ASSOCIATE WORD                       */
                  /*        16-33   PRIMARY WORD WHICH IS ALSO A CONTENT WORD            */
                  /*                                                                     */
                  /*    FORMAT OF THE OUTPUT2 DATA SET;                                  */
                  /*                                                                     */
                  /*      POSITION  FIELD DESCRIPTION                                    */
                  /*        01-05   SEQUENCE OF RECORD, FORWARD LINK TO PRIMARY WORD     */
                  /*        06-07   LENGTH OF ASSOCIATE WORD                             */
                  /*        08-12   BACKWARD LINK TO PRIMARY WORD                        */
                  /*        13-30   ASSOCIATE WORD                                       */
                  /*                                                                     */
                  /*                                                                     */
                  /*  SUGGESTED JOB CONTROL LANGUAGE:                                    */
                  /*  ++  //GO.SUF     DD DSNAME=UNC.IS.F2312.SEDELOW SUFFIX,DISP=SHR    */
                  /*      //GO.INPUT1 DD DSNAME=UNC.IS.F2312.SEDELOW.ORGTHES,DISP=SHR    */
                  /*      //GO.INPUT2 DD DSNAME=(CONTENT WORDS TO BE COMPARED),          */
                  /*      //             DISP=(OLD,PASS)                                 */
                  /*      //GO.OUTPUT1 DD DSNAME=PRIMARY,DISP=(NEW,PASS),UNIT=2314,      */
                  /*      //             SPACE=(CYL,(1,1)),VOLUME=SER=SCRTH1,            */
                  /*      //             DCB=(RECFM=FB,LRECL=42,BLKSIZE=7182,DSORG=IS,   */
                  /*      //             RPK=0,KEYLEN=9)    BLKFCTR=171                  */
                  /*      //GO.OUTPUT2 DD DSNAME=&ASSOC1,DISP=(NEW,PASS),                */
                  /*      //             UNIT=HDSK0,SPACE=(TRK,(20,10)),                 */
                  /*      //             DCB=(RECFM=FB,LRECL=30,BLKSIZE=7200)            */
                  /*                                                                     */
                  /*          ++ REQUIRED BY THE STEM SUBROUTINE.                        */
                  /*                                                                     */
                  /*                                                                     */
                  /*                                                                     */
                  /*  NEXT JOB STEP:                                                     */
                  /*                                                                     */
                  /*    THE NEXT JOB STEP IS SORT.  THE SUGGESTED JCL FOR SORT FOLLOWS:*/
                  /*                                                                     */
                  /*      1               16                                             */
                  /*      //STEP2  EXEC   PGM=IERRC000                                   */
                  /*      //SYSOUT     DD  SYSOUT=A                                      */
                  /*      //SYSPRINT   DD  SYSOUT=A                                      */
```

```
STMT LEVEL NEST

                 /*      //SYSLIB    DD  DSNAME=SYS1.SORTLIB,DISP=SHR                  */
                 /*      //SORTWK01  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTWK02  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTWK03  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTWK04  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTWK05  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTWK06  DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)           */
                 /*      //SORTIN    DD  DSNAME=&ASSOC1,DISP=(OLD,DELETE)              */
                 /*      //SORTOUT   DD  DSNAME=&ASSCC2,UNIT=HDSK0,DISP=(NEW,PASS),    */
                 /*      //              SPACE=(TRK,(20,10)),                          */
                 /*      //              DCB=(RECFM=FB,LRECL=30,BLKSIZE=7200)          */
                 /*      //SYSIN     DD  *                                             */
                 /*       SORT FIELDS=(13,18,CH,A),SIZE=E10000                         */
                 /*      /*                                                            */
                 /*                                                                    */
                 /*                                                                    */
                 /*    EXECUTE THESR02 FOLLOWING SORT.                                 */
                 /*                                                                    */
                 /*********************************************************************/

   1             THESR01:  PROCEDURE OPTIONS(MAIN);

   2     1             DECLARE
                         1 PRIMARY,
                           2 PRIMNUMC  FIXED DEC (2),  /* PRIMWD LENGTH                */
                           2 PRIMSTAT  FIXED DEC (1),  /* STATUS OF WORD               */
                           2 PRIMFREQ  FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE      */
                           2 PRMMATCNT FIXED BIN(15,0),/* MATCH COUNT                  */
                           2 FRWDLINK  FIXED BIN(15,0) /* FORWARD LINK TO ASSOCIATED   */
                               INITIAL (1),            /* WORD                         */
                           2 PRIMWD  CHAR (18);        /* PRIMARY WORD                 */

   3     1             DECLARE
                         1 ASSOCIATE,
                           2 ASSOCSEQ  FIXED BIN(15,0) /* SEQ, NEC FOR FORWARD LINK    */
                               INITIAL (1),
                           2 ASSOCNUMC FIXED DEC (2),  /* ASSOCWD LENGTH               */
                           2 BACKLINK  FIXED BIN(15,0) /* BACKWARD LINK TO PRIMARY     */
                               INITIAL (1),            /* WORD                         */
                           2 ASSOCWD CHAR (18);        /* ASSOCIATED WORD              */

   4     1             DECLARE /* INPUT FROM 'SUFFIX'                                  */
                         1 CONTENTWD,
                           2 NUMCCNTWD FIXED DEC (2),  /* CONTENT WORD LENGTH          */
                           2 MATCNTWD  FIXED BIN(15,0),/* MATCH COUNT                  */
                           2 FREQCNTWD FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE      */
                           2 CNTWD   CHAR (18);        /* CONTENT WORD                 */

   5     1             DECLARE
                        SAME_ROOT     FIXED DEC (1),  /* SWITCH TO DENOTE COMMON ROOT*/
```

```
          /* THESR01:  PROCEDURE TO COMPARE CONTENT WORDS TO  PRIMARY WORDS.   */

STMT LEVEL NEST

                          SAVEPRIM     CHAR (58) VARYING,/* PRIMWD SAVE AREA              */
                          BLKWK        CHAR (18) VARYING /* WORK READ IN AREA             */
                                INITIAL ('');

  6     1               DECLARE
                          OUTPUT1 FILE ENVIRONMENT (INDEXED) SEQUENTIAL KEYED;

  7     1               OPEN FILE(INPUT1)  INPUT;   /* PRIMARY AND ASSOCIATED WORDS    */
  8     1                 ON ENDFILE(INPUT1) GO TO FINISH;

 10     1               OPEN FILE(INPUT2) INPUT;   /* INPUT FROM SUFFIX               */
 11     1                 ON ENDFILE(INPUT2) GO TO RDPRIM;

 13     1               OPEN FILE(OUTPUT1) OUTPUT, FILE(OUTPUT2) OUTPUT;

 14     1         RDPRIM:
                        GET FILE(INPUT1) EDIT (PRIMNUMC,PRIMWD)
                                (F(2),A(18));

 15     1         RDTEXT:
                        GET FILE(INPUT2) EDIT (CONTENTWD,BLKWK) (F(2),F(5),F(5),
                                A(NUMCCNTWD),A(18-NUMCCNTWD));

 16     1         COMPARE:
 17     1               IF CNTWD = PRIMWD THEN DO;
 18     1                 PRIMSTAT = 0;
 19     1                 GO TO SAME;
 20     1                 END;

 21     1               IF SUBSTR(CNTWD,1,3) = SUBSTR(PRIMWD,1,3) THEN GO TO CKROOT;

 23     1         CKSEQ:
 24     1               IF CNTWD < PRIMWD THEN GO TO RDTEXT;

                  /*    IF CNTWD > PRIMWD THEN DC;   *** DEFAULT ***                   */

 25     1               SAVEPRIM = PRIMWD;
 26     1               IF PRIMSTAT = 0 THEN GO TO SKIPRCD;
 28     1               PRIMFREQ = 0;
 29     1               PRMMATCNT = C;
 30     1               GO TC WRITEPRIM;
                  /*      END;           ***** END DEFAULT ***                         */

 31     1         SKIPRCD:
                        GET FILE(INPUT1) EDIT (BLKWK) (X(21),A(1));  /* SKIP ASSOC WORD*/
 32     1               GET FILE(INPUT1) EDIT (PRIMNUMC,PRIMWD)
                                (F(2),A(18));
 33     1               IF SAVEPRIM = PRIMWD THEN GO TC SKIPRCD;
 35     1               GO TC COMPARE;
```

```
        /* THESR01:  PROCEDURE TO COMPARE CONTENT WORDS TO  PRIMARY WORDS.   */                          PAGE      6

STMT LEVEL NEST

 36      1          SAME:
                          SAVEPRIM = PRIMWD;
 37      1                PRIMFREQ = FREQCNTWD;
 38      1                PRMMATCNT = MATCNTWD;
 39      1                PRIMSTAT = 0;

 40      1          WRITEPRIM:                                  /********************************/
                          GET FILE(INPUT2) EDIT                 /*                              */
                                  (CONTENTWD,BLKWK)             /* CHECK FOR CONTENT WORD WITH */
                                  (F(2),F(5),F(5),              /* SAME MATCH COUNT.  TOTAL     */
                                  A(NUMCCNTWD),                 /* FREQUENCY OF OCCURRENCE FOR */
                                  A(18-NUMCCNTWD));             /* ALL WORDS WITH SAME MATCH    */
                                                                /* COUNT.  INDICATE IN          */
 41      1                IF PRMMATCNT = MATCNTWD THEN DO;/* 'PRIMSTAT' BYTE THAT THE    */
 43      1                  PRIMFREQ = PRIMFREQ +               /* WORD OCCURS IN MORE THAN     */
                                  FREQCNTWD;                    /* ONE FORM IN TEXT.            */
 44      1                  SAVEPRIM = PRIMWD;                  /*                              */
 45      1                  PRIMSTAT = 1;                       /********************************/
 46      1                  GO TO WRITEPRIM;
 47      1                  END;

 48      1                WRITE FILE(OUTPUT1) FROM (PRIMARY) KEYFROM (BACKLINK);

 49      1          WRITEASSOC:
                          GET FILE(INPUT1) EDIT (ASSOCNUMC,ASSOCWD)
                                  (F(2),A(18));
 50      1                PUT FILE(OUTPUT2) EDIT (ASSOCIATE) (F(5),F(2),F(5),A(18));
 51      1                ASSOCSEQ = ASSOCSEQ + 1;
 52      1                FRWDLINK = FRWDLINK + 1;
 53      1                GET FILE(INPUT1) EDIT (PRIMNUMC,PRIMWD)
                                  (X(1),F(2),A(18));
 54      1                IF SAVEPRIM = PRIMWD THEN GO TO WRITEASSOC;
 56      1                BACKLINK = BACKLINK + 1;
 57      1                GO TC CCMPARE;

 58      1          CKROOT:
                                                                /********************************/
                                                                /*                              */
                          CALL STEM (SAME_ROOT,PRIMNUMC,        /* CHECK FOR POSSIBLE COMMON    */
                              PRIMWD,NUMCCNTWD,CNTWD);          /* ROOT BASED ON SUFFICES       */
                                                                /*                              */
                                                                /********************************/

 59      1                IF SAME_ROOT = 1 THEN DO;
 61      1                  PRIMSTAT = 1;
 62      1                  GO TO SAME;
 63      1                  END;
 64      1                GO TC CKSEQ;
```

```
        /* THESR01:  PROCEDURE TO COMPARE CONTENT WORDS TO  PRIMARY WORDS.   */                    PAGE      7

 STMT LEVEL NEST

  65     1         FINISH:
                         PRIMWD = (18) '9';
  66     1               BACKLINK = BACKLINK + 1;
  67     1               WRITE FILE (OUTPUT1) FROM (PRIMARY) KEYFROM (BACKLINK);

  68     1               CLOSE FILE(INPUT1), FILE(INPUT2), FILE(OUTPUT1),
                               FILE(OUTPUT2);

  69     1               END THESR01;
```

```
STMT LEVEL NEST
                 /*  'THESR02': PROCEDURE TO COMPARE ASSOCIATE WORDS TO CONTENT WORDS */

                 /*********************************************************************/
                 /*                                                                  */
                 /*  GENERAL PROGRAP rLOW:                                           */
                 /*                                                                  */
                 /*  THE THESR02 PRCGRAM COMPARES ASSOCIATE WORDS TO CONTENT WORDS   */
                 /*  SETTING THE MATCH COUNT AND FREQUENCY COUNT OBTAINED FROM THE   */
                 /*  CONTENT WORD RECORDS WHEN A MATCH IS FOUND.  IF THE ASSOCIATE   */
                 /*  WORD APPEARS IN ANOTHER FORM IN THE TEXT, THE 'ASSOCSTAT' FIELD */
                 /*  IS SET TO THF VALUE 1.  WHEN NO MATCH HAS BEEN FOUND THE        */
                 /*  'ASSOCMAT' AND 'ASSOCFREQ' FIELDS ARE SET TO ZERO.              */
                 /*  THE ASSOCIATE WORD FILE MUST BE IN ASCENDING SEQUENCE ON THE    */
                 /*  ASSOCIATE WORD PRIOR TO INTRODUCTION TO THIS PROGRAM.  THE      */
                 /*  CONTENT WORD FILE IS IN MATCH CCUNT ORDER.                      */
                 /*                                                                  */
                 /*                                                                  */
                 /*  INPUT:                                                          */
                 /*                                                                  */
                 /*    ASSOCIATE WORD RECORD FCRMAT (INPUT1);                        */
                 /*                                                                  */
                 /*      THE USER IS REFERRED TO THE THESR01 PRCGRAM OUTPUT2         */
                 /*      DOCUMENTATION.                                              */
                 /*                                                                  */
                 /*    CONTENT WORD RECORD FCRMAT (INPUT2);                          */
                 /*                                                                  */
                 /*      THE USER IS REFERRED TO THE THESR01 PRCGRAM INPUT2          */
                 /*      DOCUMENTATION.                                              */
                 /*                                                                  */
                 /*                                                                  */
                 /*  OUTPUT:                                                         */
                 /*                                                                  */
                 /*    FORMAT OF THE OUTPUT1 DATA SET;                               */
                 /*                                                                  */
                 /*      POSITION   FIELD DESCRIPTION                                */
                 /*        01-05    SEQUENCE OF RECORD, FORWARD LINK TO PRIMARY WORD */
                 /*        06-C7    LENGTH OF ASSOCIATE WORD                         */
                 /*           08    STATUS OF WORD.  EQUALS 1 IF THE WORD APPEARS IN A */
                 /*                           DIFFERENT FORM IN THE TEXT AND 0       */
                 /*                           OTHERWISE.                             */
                 /*        09-13    BACKWARD LINK TO PRIMARY WORD                    */
                 /*        14-18    MATCH COUNT                                      */
                 /*        19-23    FREQUENCY COUNT                                  */
                 /*        24-41    ASSOCIATE WORD                                   */
                 /*                                                                  */
                 /*                                                                  */
                 /*  SUGGESTED JOB CONTROL LANGUAGE                                  */
                 /*                                                                  */
                 /*     //GO.SUF       DD DSNAME=UNC.1..F2312.SEDELOW.SUFFIX,DISP=SHR */
                 /*     //GO.INPUT1    DD DSNAME=&ASSOC2,DISP=(OLD,DELETE)           */
                 /*     //GO.INPUT2    DD DSNAME=CCNTENT-WORDS-TO-BE-COMPARED,       */
                 /*     //                DISP=(OLD,PASS),UNIT=XXXX,VOLUME=SER=XXXXXX */
```

```
STMT LEVEL NEST

                    /*      //GO.OUTPUT1 DD DSNAME=&ASSCC3,DISP=(NFW,PASS),               */
                    /*      //                 SPACE=(TRK,(20,10)),UNIT=HDSKO,            */
                    /*      //                 DCB=(RECFM=FB,LRECL=41,BLKSIZE=7175)       */
                    /*                                                                    */
                    /*                                                                    */
                    /*  NEXT JCB STEP:                                                    */
                    /*                                                                    */
                    /*      EXECUTE THESR03.                                              */
                    /*                                                                    */
                    /*********************************************************************/
   1                THESR02:  PROCEDURE OPTIONS (MAIN);

   2      1               DECLARE
                            1 ASSOCIATE,
                              2 FRWDLINK  CHAR (5),         /* ORIGINAL SEQ, FORWARD LINK  */
                              2 ASSOCNUMC FIXED DEC (2),   /* WORD LENGTH                 */
                              2 ASSOCSTAT FIXED DEC (1),   /* WORD STATUS                 */
                              2 BACKLINK  FIXED BIN(15,0),/* BACKWARD LINK TO PRIMARY    */
                              2 ASSOCMAT  FIXED BIN(15,0),/* MARCH COUNT                 */
                              2 ASSOCFFEQ FIXED BIN(15,0),/* FREQUENCY COUNT             */
                              2 ASSOCWD   CHAR (18);       /* WORD                        */
   3      1               DECLARE
                            1 CONTENTWD,
                              2 NUMCCNTWD FIXED DEC (2),   /* LENGTH OF CONTENT WORD      */
                              2 MATCNTWD  FIXED BIN(15,0),/* PATCH COUNT                 */
                              2 FREQCNTWD FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE     */
                              2 CNTWD   CHAR (58) VARYING;/* CONTENT WORD                */
   4      1               DECLARE
                            SAME_ROOT     FIXED DEC (1),   /* SWITCH TO DENOTE COMMON ROOT*/
                            BLKWK  CHAR (18) VARYING,      /* WORK READIN AREA            */
                            SAVELGTH      FIXED DEC (2),   /* LENGTH OF ASSOCIATE WORD    */
                            SAVEASSOC   CHAR (58) VARYING;/* ASSOCWD SAVE AREA           */

   5      1               OPEN FILE(INPUT1) INPUT;        /* ASSOCIATE WORDS             */
   6      1                 ON ENDFILE(INPUT1) GO TO FINISH;

   8      1               OPEN FILE(INPUT2) INPUT;        /* INPUT FROM SUFFIX           */
   9      1                 ON ENDFILE(INPUT2) GO TO RDASSOC;

  11      1         RDASSOC:
                          GET FILE(INPUT1) EDIT (FRWDLINK,ASSOCNUMC,ASSOCSTAT,BACKLINK,
                                  ASSOCWD) (A(5),F(2),F(1),F(5),A(18));

  12      1         RDTEXT:
                          GET FILE(INPUT2) EDIT (CONTENTWD,BLKWK)
```

```
STMT LEVEL NEST
                                 (F(2),F(5),F(5),A(NUMCCNTWD),A(18-NUMCCNTWD));

  13     1           COMPARE:
  14     1                 IF CNTWD = ASSOCWD THEN GO TO SAME;
  15     1                 IF SUBSTR(CNTWD,1,3) = SUBSTR(ASSOCWD,1,3) THEN GO TO CKROOT;
  17     1           CKSEQ:
  18     1                 IF CNTWD < ASSOCWD THEN GO TO RDTEXT;
                     /*    IF CNTWD > ASSOCWD THEN DO;   *** DEFAULT ***                    */
  19     1                   ASSOCSTAT = 0;
  20     1                   ASSOCFREQ = 0;
  21     1                   ASSOCMAT = 0;
  22     1                 GO TO WRITEASSOC;
                     /*     END;          **** DEFAULT END ****                             */

  23     1           SAME:
                           SAVEASSOC = ASSOCWD;
  24     1                 SAVELGTH = ASSOCNUMC;
  25     1                 ASSCCSTAT = 0;
  26     1                 ASSOCFREQ = FREQCNTWD;
  27     1                 ASSOCMAT = MATCNTWD;

  28     1           WRITEASSOC:
                           GET FILE(INPUT2) EDIT (CONTENTWD,BLKWK)
                                 (F(2),F(5),F(5),A(NUMCCNTWD),A(18-NUMCCNTWD));
  29     1                 IF ASSOCMAT = MATCNTWD THEN DO;
  31     1                   ASSOCFREQ = ASSOCFREQ + FREQCNTWD;
  32     1                   ASSOCSTAT = 1;
  33     1                   GO TO WRITEASSOC;
  34     1                   END;

  35     1           PUTASSOC:
                           PUT FILE(OUTEUT1) EDIT (ASSOCIATE) (A(5),F(2),F(1),F(5),F(5),
                                F(5),A(18));
  36     1                 GET FILE(INPUT1) EDIT (FRWDLINK,ASSCCNUMC,BACKLINK,ASSOCWD)
                                 (A(5),F(2),X(1),F(5),A(18));

  37     1                 IF SAVEASSOC = ASSOCWD THEN GO TO PUTASSCC;
  39     1                 IF SUBSTR(SAVEASSCC,1,3) = SUESTR(ASSOCWD,1,3) THEN DO;
  41     1                   CALL STEM (SAME_ROOT,SAVELGTH,SAVEASSOC,ASSOCNUMC,ASSOCWD);
  42     1                   IF SAME_ROCT = 1 THEN GO TO PUTASSOC;
  44     1                   END;

  45     1                 GO TO COMPARF;

  46     1           CKROOT:
                           CALL STEM (SAME_ROOT,NUMCCNTWD,CNTWD,ASSOCNUMC,ASSOCWD);
  47     1                 IF SAME_ROOT = 1 THEN GO TO SAME;
  49     1                 GO TO CKSEQ;

  50     1           FINISH:
```

```
STMT LEVEL NEST

                   CLOSE FILE(INPUT1), FILE(INPUT2), FILE(OUTPUT1);
 51     1          END THESR02;
```

```
STMT LEVEL NEST
                  /* 'THESR03':  PROCEDURE TO COMPARE ASSOCIATE WORDS TO PRIMARY WORDS */

                  /*********************************************************************/
                  /*                                                                   */
                  /*  GENERAL PROGRAM FLOW:                                            */
                  /*                                                                   */
                  /*  THE THESR03 PROGRAM COMPARES ASSOCIATE WORDS AND PRIMARY WORDS,  */
                  /*  SETTING A FORWARD LINK TO THE PRIMARY WORD WHEN A MATCH OCCURS.  */
                  /*  WHEN NO MATCH HAS BEEN FOUND THE FIELD 'PRIMLINK' WILL BE EQUAL  */
                  /*  TO ZERO.                                                         */
                  /*                                                                   */
                  /*  THE PRIMARY ANL ASSOCIATE WORD FILES MUST BE IN ASCENDING        */
                  /*  ALPHAEETIC SEQUENCE.                                             */
                  /*                                                                   */
                  /*                                                                   */
                  /*  INPUT:                                                           */
                  /*                                                                   */
                  /*    INPUT FORMAT FOR PRIMARY WORD DATA SET (INPUT1);               */
                  /*                                                                   */
                  /*      THE USER IS REFERRED TO THE THESR01 PROGRAM OUTPUT1          */
                  /*      DOCUMENTATION.                                               */
                  /*                                                                   */
                  /*    INPUT FORMAT FOR THE ASSOCIATE WORD DATA SET (INPUT2);         */
                  /*                                                                   */
                  /*      THE USER IS REFERRED TO THE THESR02 PROGRAM OUTPUT1          */
                  /*      DCCUMENTATION.                                               */
                  /*                                                                   */
                  /*      THE ASSOCIATE WORD DATA SET MUST BE IN ASCENDING SEQUENCE ON */
                  /*      THE WORDS.  OUTPUT FROM THE THESR02 PROGRAM IS IN CORRECT    */
                  /*      SEQUENCE FOR INTRODUCTION TO THIS PROGRAM.                   */
                  /*                                                                   */
                  /*                                                                   */
                  /*  OUTPUT:                                                          */
                  /*                                                                   */
                  /*    FORMAT OF THE OUTPUT1 DATA SET;                                */
                  /*                                                                   */
                  /*      PCSITION  FIELD DESCRIPTION                                  */
                  /*        01-05   CRIGINAL SEQUELCE                                  */
                  /*        06-07   IENGTH OF WORD                                     */
                  /*           08   WORD STATUS                                        */
                  /*           09   PREVIOUS APPEARANCE INDICATION.  VALUE EQUALS      */
                  /*                          ONE IF THE WORD APPEARS IN PREVIOUS TEXT */
                  /*        10-14   EACKWARD LINK TO PRIMARY WORD                      */
                  /*        15-19   MATCH COUNT                                        */
                  /*        20-24   FREQUENCY OF OCCURRENCE                            */
                  /*        25-29   SECONDARY LINK TO PRIMARY WORD                     */
                  /*        30-47   ASSOCIATE WORD                                     */
                  /*                                                                   */
                  /*                                                                   */
                  /*      THIS OUTPUT DATA SET SHOULD BE SAVED.  IT IS THIS DATA SET   */
                  /*      FROM AN EARLIER JOB WHICH IS INTORDUCED TO THE THESR04       */
                  /*      PROGRAM ALONG WITH THE CURRENT OUTPUT1 FILE.                 */
/*                                                                          */
/*                                                                          */
/*  SUGGESTED JOB CONTROL LANGUAGE:                                         */
/*                                                                          */
/*  ++  //GO.SUP     DD DSNAME=UNC.IS.F2312.SEDELOW.SUFFIX,DISP=SHR         */
/*      //GO.INPUT1 DD DSNAME=PRIMARY,DISP=(OLD,PASS)                       */
/*      //GO.INPUT2 DD DSNAME=&ASSCC3,DISP=(OLD,DELETE)                     */
/*      //GO.OUTPUT1 DD DSNAME=&ASSOC4,DISP=(NEW,PASS),UNIT=HDSK0,          */
/*      //              SPACE=(TRK,(20,10)),                                */
/*      //              DCB=(RECFM=FB,LRECL=47,BLKSIZE=7191)                */
/*                                                                          */
/*        ++ REQUIRED FOR THE STEM SUBROUTINE                               */
/*                                                                          */
/*  NEXT JOB STEP:                                                          */
/*                                                                          */
/*    EXECUTE THESR04.  THIS STEP IS A USER OPTION.  THESR04                */
/*    COMPARES THE CURRENT FILE OF ASSOCIATE WORDS WITH A FILE OF           */
/*    ASSOCIATE WOFDS PRODUC   ON AN EARLIER RUN AND INDICATES WHEN         */
'*    THE WORD HAS APPEARED IN THE EARLIER FILE BY SETTING THE              */
/*    FIELD 'ASSOCAPPR' EQUAL TC THE VALUE ONE.                             */
/*                                                                          */
/*    IF THE USER ELECTS NOT TO EXECUTE THESR04 THE NEXT JOB STEP IS        */
/*    SORT.  THE SUGGESTED JCL FOR SORT FOLLOWS:                            */
/*                                                                          */
/*      1              16                                                   */
/*      //STEP5  EXEC   PGM=IERRC000                                        */
/*      //SYSOUT     DD  SYSOUT=A                                           */
/*      //SYSPRINT   DD  SYSOUT=A                                           */
/*      //SYSLIB     DD  DSNAME=SYS1.SORTLIB,DISP=SHR                       */
/*      //SORTWK01   DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTWK02   DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTWK03   DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTWK04   CD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTWK05   DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTWK06   DD  UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)                */
/*      //SORTIN     DD DSNAME=&ASSOC4,DISP=(OLD,DELETE)                    */
/*      //SORTOUT    DD DSNAME=&ASSCC6,UNIT=HDSK0,DISP=(NEW,PASS),          */
/*      //               SPACE=(TRK,(20,10)),                               */
/*      //              DCB=(RECFM=FB,LRECL=47,BLKSIZE=7191)                */
/*      //SYSIN     DD  *                                                   */
/*       SORT FIELDS=(1,5,CH,A),SIZE=E10000                                 */
/*      /*                                                                  */
/*                                                                          */
/*                                                                          */
/*    EXECUTE THESR05 FOLLOWING SORT.                                       */
/*                                                                          */
/**************************************************************************/

THESR03:  PROCEDURE OPTIONS (MAIN);
```

```
STMT LEVEL NEST

  2     1             DECLARE
                        1 PRIMARY,
                          2 PRIMNUMC   FIXED DEC (2),  /* PRIMWD LENGTH                */
                          2 PRIMSTAT   FIXED DEC (1),  /* STATUS OF WORD               */
                          2 PRIMFREQ   FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE      */
                          2 PRMMATCNT  FIXED BIN(15,0),/* MATCH COUNT                  */
                          2 FRWDLINK   FIXED BIN(15,0),/* FORWARD LINK TO ASSOC WORD   */
                          2 PRIMWD     CHAR (18);      /* PRIMARY WORD                 */

  3     1             DECLARE
                        1 ASSOCIATE,
                          2 ASSOCSEQ   CHAR (5),       /* ORIGINAL SEQUENCE            */
                          2 ASSOCNUMC  FIXED DEC (2),  /* LENGTH OF WORD               */
                          2 ASSOCSTAT  FIXED DEC (1),  /* WORD STATUS                  */
                          2 ASSOCAPPR  FIXED DEC (1)   /*  1 IF WORD APPEARS IN        */
                              INITIAL (0),             /*  PREVIOUS TEXT               */
                          2 BACKLINK   FIXED BIN(15,0),/* BACKWARD LINK TO PRIMARY     */
                          2 ASSOCMAT   FIXED BIN(15,0),/* MATCH COUNT                  */
                          2 ASSOCFREQ  FIXED BIN(15,0),/* FREQUCNCY OF OCCURRENCE      */
                          2 PRIMLINK   FIXED BIN(15,0),/* SECONDARY LINK TO PRIMARY W:*/
                          2 ASSOCWD    CHAR (18);      /* ASSOCIATE WORD               */

  4     1             DECLARE
                        LOCSEQ         FIXED BIN(15,0) /* SEQUENCE OF PRIMARY WORDS    */
                              INITIAL (0);

  5     1             DECLARE
                        INPUT1  FILE SEQUENTIAL KEYED ENVIRONMENT (INDEXED);

  6     1             OPEN FILE(INPUT1) INPUT;       /* PRIMARY WORDS INPUT           */

  7     1             OPEN FILE(INPUT2) INPUT;       /* ASSOCIATE WORDS INPUT         */
  8     1               ON ENDFILE(INPUT2) GO TO FINISH;

 10     1             OPEN FILE(OUTPUT1) OUTPUT;     /* ASSOCIATE WORD OUTPUT         */


                /*********************************************************************/
                /*                                                                   */
                /*  ON THE END-OF-FILE CONDITION, COPY THE REMAINING ASSOCIATE       */
                /*  RECORDS ONTO OUTPUT1.                                            */
                /*                                                                   */
                /*********************************************************************/

 11     1             ON ENDFILE(INPUT1) BEGIN;
 13     2               PRIMLINK = 0;
```

```
        /* 'THESR03':   PROCEDURE TO COMPARE ASSOCIATE WORDS TO PRIMARY WORDS */                                PAGE     5

 STMT LEVEL NEST

  14     2            PUTASSOC3:
                              PUT FILE(OUTPUT1) EDIT (ASSOCIATE)  (A(5),F(2),F(1),F(1),F(5),
                                    F(5),F(5),F(5),A(18));
  15     2                    GET FILE(INPUT2) EDIT (ASSOCSEQ,ASSOCNUMC,ASSOCSTAT
                                    BACKLINK,ASSOCMAT,ASSOCFREQ,ASSOCWD)
                                    (A(5),F(2),F(1),F(5),F(5),F(5),A(18));
  16     2                    GC TO PUTASSOC3;
  17     2                    END;


  18     1            RDPRM:
                           READ FILE(INPUT1) INTO (PRIMARY);
  19     1                 LOCSEQ = LOCSEQ + 1;

  20     1            RDASSOC:
                           GET FILE(INPUT2) EDIT (ASSOCSEQ,ASSOCNUMC,ASSOCSTAT,BACKLINK,
                                   ASSOCMAT,ASSOCFREQ,ASSOCWD)  (A(5),F(2),F(1),F(5),
                                   F(5),F(5),A(18));

  21     1            COMPARE:
  22     1                 IF PRIMWD = ASSOCWD THEN GO TO PUTASSOC1;
  23     1                 IF SUBSTR(PRIMWD,1,3) = SUBSTR(ASSOCWD,1,3) THEN DO;
  25     1                   CALL STEM (SAME_ROOT,PRIMNUMC,PRIMWD,ASSOCNUMC,ASSOCWD);
  26     1                   IF SAME_ROOT = 1 THEN GO TC PUTASSOC1;
  28     1                   END;

  29     1                 IF PRIMWD > ASSOCWD THEN DO;
  31     1                   PRIMLINK = 0;
  32     1                   GC TO PUTASSOC2;
  33     1                   END;

                      /*   IF PRIMWD < ASSOCWD THEN DO;    **** DEFAULT ****                  */
  34     1                   READ FILE(INPUT1) INTC (PRIMARY);
  35     1                   LOCSEQ = LCCSEQ + 1;
  36     1                   GO TO COMPARE;
                      /*     END;                                                              */

  37     1            PUTASSOC1:
                           PRIMLINK = LCCSEQ;

  38     1            PUTASSOC2:
                           PUT FILE(OUTPUT1) EDIT (ASSOCIATE)  (A(5),F(2),F(1),F(1),F(5),
                                   F(5),F(5),F(5),A(18));

  39     1                 GO TO RDASSOC;

  40     1            FINISH:
                           CLCSE FILE(INPUT1), FILE(INPUT2), FILE(OUTPUT1);

  41     1                 *END THESR03;
```

```
STMT LEVEL NEST
                  /* THESR04:  PROCEDURE TO COMPARE ASSOCIATE WORDS OF DIFFERENT TEXTS */

                  /**********************************************************************/
                  /*                                                                    */
                  /*  GENERAL FLOW.                                                     */
                  /*                                                                    */
                  /*  THE THESR04 PROGRAM IS EXECUTED BY USER OPTION.  THE PROGRAM      */
                  /*  COMPARES THE CURRENT ASSOCIATE WORD DATA SET WITH ONE             */
                  /*  PRODUCED DURING AN EARLIER RUN.  WHEN A MATCH OCCURS THE FIELD    */
                  /*  'ASCAPPRONE' IS SET EQUAL TO THE VALUE ONE.  THIS FIELD IS        */
                  /*  TESTED DURING THE EXECUTION OF THESR06 AND AN APPROPRIATE         */
                  /*  FLAG PRINTED TO INDICATE THE MATCH.                               */
                  /*                                                                    */
                  /*                                                                    */
                  /*  INPUT:                                                            */
                  /*                                                                    */
                  /*    CURRENT ASSOCIATE WORD DATA SET (INPUT1);                       */
                  /*                                                                    */
                  /*      THE USER IS REFERRED TO THE THESR03 OUTPUT1 RECORD FORMAT.    */
                  /*                                                                    */
                  /*    PREVIOUS ASSCCIATE WORD DATA SET (INPUT2);                      */
                  /*                                                                    */
                  /*      THE USER IS REFERRED TO THE THESR03 OUTPUT1 RECORD FORMAT     */
                  /*                                                                    */
                  /*                                                                    */
                  /*  OUTPUT:                                                           */
                  /*                                                                    */
                  /*    OUTPUT1 RECORD FORMAT;                                          */
                  /*                                                                    */
                  /*      THE INCOMING RECORD FORMAT IS NOT ALTERED.                    */
                  /*                                                                    */
                  /*                                                                    */
                  /*  SUGGESTED JOB CONTROL LANGUAGE:                                   */
                  /*                                                                     /
                  /*      //GO.SUP     DD DSNAME=UNC.IS.F2312.SEDELOW.SUFFIX,DISP=SHR   */
                  /*      //GO.INPUT1 DD DSNAME=&ASSOC4,DISP=(OLD,KEEP)                 */
                  /*      //GO.INPUT2 DD DSNAME=PREVIOUS-ASSOCIATE-WORD-FILE,           */
                  /*      //              DISP=(OLD,KEEP)                               */
                  /*      //GO.OUTPUT1 DD DSNAME=&ASSCC5,DISP=(NEW,PASS),               */
                  /*      //              DCB=(RECFM=FB,LRECL=47,BLKSIZE=7191),         */
                  /*      //              SPACE=(TRK,(20,10))                           */
                  /*                                                                    */
                  /*                                                                    */
                  /*  NEXT JOB STEP:                                                    */
                  /*                                                                    */
                  /*    THE NEXT JOB STEP IS SORT.  THE SUGGESTED JCL FOLLOWS;          */
                  /*                                                                    */
                  /*      1             16                                              */
                  /*      //STEP5  EXEC  PGM=IERRCO00                                   */
                  /*      //SYSOUT     DD SYSOUT=A                                      */
                  /*      //SYSPRINT   DD SYSOUT=A                                      */
                  /*      //SYSLIB     DD DSNAME=SYS1.SORTLIB,DISP=SHR                  */
```

```
STMT LEVEL NEST

                    /*      //SORTWK01  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTWK02  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTWK03  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTWK04  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTWK05  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTWK06  DD UNIT=HDSK0,SPACE=(TRK,(50),,CONTIG)          */
                    /*      //SORTIN    DD DSNAME=&ASSOC5,DISP=(OLD,DELETE)             */
                    /*      //SORTOUT   DD DSNAME=&ASSOC6,UNIT=HDSK0,DISP=(NEW,PASS),   */
                    /*      //             SPACE=(TRK,(20,10)),                         */
                    /*      //             DCB=(RECFM=FB,LRECL=47,BLKSIZE=7191)         */
                    /*      //SYSIN     DD *                                            */
                    /*       SORT FIELDS(1,5,CH,A),SIZE=E10000                          */
                    /*      /*                                                          */
                    /*                                                                  */
                    /*    EXECUTE THESR05 FOLIOWING SOFT.                               */
                    /*                                                                  */
                    /********************************************************************/

   1                THESR04:  PROCEDURE OPTIONS (MAIN);

   2       1              DECLARE
                            1 ASCONE,
                              2 ASCSEQ     CHAR (5),         /* ORIGINAL SEQUENCE           */
                              2 ASCNUMCONE FIXED DEC (2), /* LENGTH OF ASSOC WORD ONE    */
                              2 ASCSTAT    CHAR (1),         /* STATUS OF WORD              */
                              2 ASCAPPRONE FIXED DEC (1), /* WORD APPEARANCE IN ASSOC ONE*/
                              2 WORKINCNE  CHAR (20),        /* BACKLINK ETC.               */
                              2 ASCWDONE   CHAR (18);        /* ASSOCIATE WORD ONE          */

   3       1              DECLARE
                            1 ASCTWO,
                              2 ASCSEQTWO  CHAR (5),         /* ORIGINAL SEQUENCE           */
                              2 ASCNUMCTWO FIXED DEC (2)     /* LWNTHE OF WORD TWO          */
                              2 WORKINTWO  CHAR (22),        /* ASSOCSTAT, ETC.             */
                              2 ASCWDTWO   CHAR (18);        /* SECOND ASSOC WORD           */

   4       1              DECLARE
                            SAME_ROOT      FIXED DEC (1);    /* SAME ROOT INDICATOR         */.

   5       1              OPEN FILE(INPUT1) INPUT;           /* CURRENT ASSOC WD FILE       */
   6       1              ON ENDFILE(INPUT1) GO TO FINISH;

   8       1              OPEN FILE(INPUT2) INPUT;           /* PREVIOUS ASSOC WD FILE      */

                    /*******************************************************************/
                    /*                                                                 */
                    /*  ON END-OF-FILE CONDITION COPY REMAINING PORTION OF FILE.       */
                    /*                                                                 */
                    /*******************************************************************/
```

```
        /* THESR04:  PROCEDURE TO COMPARE ASSOCIATE WORDS OF DIFFERENT TEXTS */                          PAGE    4

 STMT LEVEL NEST

   9    1                 ON ENDFILE(INPUT2) BEGIN;

  11    2           PUTASSOC1:
                            PUT FILE(OUTPUT1) EDIT (ASCONE)  (A(5),F(2),A(1),F(1),A(20),
                                  A(18));
  12    2                   GET FILE(INPUT1) EDIT (ASCONE)  (A(5),F(2),A(1),F(1),A(20),
                                  A(18));
  13    2                   GO TO PUTASSOC1;
  14    2                   END;


  15    1                 OPEN FILE(OUTPUT1) OUTPUT;      /* UPDATED ASSOC WD FILE

  16    1           RDPRM:
                          GET FILE(INPUT2) EDIT (ASCTWO)  (A(5),F(2),A(22),A(18));

  17    1           RDSEC:
                          GET FILE(INPUT1) EDIT (ASCONE)  (A(5),F(2),A(1),F(1),A(20),
                                  A(18));

  18    1           COMPARE:
  19    1                 IF ASCWDONE = ASCWDTWO THEN DO;
  20    1                   ASCAPPRONE = 1;
  21    1                   GO TO PUTASSOC2;
  22    1                   END;
  23    1                 IF SUBSTR(ASCWDONE,1,3) = SUBSTR(ASCWDTWO,1,3) THEN DO;
  25    1                   CALL STEM (SAME_ROOT,ASCNUMCONE,ASCWDONE,ASCNUMCTWO,ASCWDTWO);
  26    1                   IF SAME_ROOT = 1 THEN DO;
  28    1                     ASCAPPRONE = 1;
  29    1                     GO TO PUTASSOC2;
  30    1                     END;
  31    1                   END;

  32    1                 IF ASCWDONE < ASCWDTWO THEN GO TO PUTASSOC2;

                    /*    IF ASCWDONE > ASCWDTWO THEN DO; DEFAULT                               */
  34    1                   GET FILE(INPUT2) EDIT (ASCTWO)  (A(5),F(2),A(22),A(18));
  35    1                   GO TO COMPARE;
                    /*      END;                            DEFAULT                             */

  36    1           PUTASSOC2:
                          PUT FILE(OUTEUT1) EDIT (ASCONE)(A(5),F(2),A(1),F(1),A(20),A(18));
  37    1                 GO TO RDSEC;


  38    1           FINISH:
                          CLCSE FILE(INPUT1), FILE(INPUT2), FILE(OUTPUT1);

  39    1                 END THESR04;
```

```
STMT LEVEL NEST
                  /* 'THESR05':  PROCEDURE TO WRITE ASSOCIATE WORD DATA SET AFTER SORT */

                  /*********************************************************************/
                  /*                                                                   */
                  /*   GENERAL FLOW                                                    */
                  /*                                                                   */
                  /*   THE PRCGRAM WRITES AS AN INDEXED SEQUENTIAL DATA SET THE        */
                  /*   ASSOCIATE WORD RECORDS.                                         */
                  /*                                                                   */
                  /*   THE ASSOCIATE WORD DATA SET MUST BE SORTED ON THE ORIGINAL      */
                  /*   SEQUENCE FIELD PRIOR TO INTRODUCTION TO THIS PROGRAM.           */
                  /*                                                                   */
                  /*                                                                   */
                  /*   INPUT:                                                          */
                  /*                                                                   */
                  /*     INPUT1 DATA SET;                                              */
                  /*                                                                   */
                  /*       THE USER IS REFERRED TO THE THESRC3 OUTPUT1 DOCUMENTATION.  */
                  /*                                                                   */
                  /*                                                                   */
                  /*   OUTPUT:                                                         */
                  /*                                                                   */
                  /*     OUTPUT1 DATA SET;                                             */
                  /*                                                                   */
                  /*       POSITION   FIELD DESCRIPTION                                */
                  /*         01-05    CRIGINAL SEQUENCE                                */
                  /*         06-07    IENGTH OF WORD                                   */
                  /*            08    WORD STATUS,  EQUAI TO THE VALUE ONE IF WORD     */
                  /*                            APPEARS IN DIFFERENT FORM IN TEXT.     */
                  /*            09    PREVIOUS APPEARANCE.  EQUAL TO THE VALUE ONE     */
                  /*                            IF THE WORD PREVIOUSLY APPEARED IN AN  */
                  /*                            EARLIER PROCESSED  PORTION OF THE TEXT. */
                  /*         10-13    BACKWARD LINK TO PRIMARY WORD                    */
                  /*         14-17    MATCH COUNT                                      */
                  /*         18-21    FREQUENCY OF OCCURRENCE                          */
                  /*         22-25    SECONDARY LINK TO PRIMARY WORD                   */
                  /*         26-43    ASSOCIATE WORD                                   */
                  /*                                                                   */
                  /*                                                                   */
                  /*   SUGGESTED JOB CONTROL LANGUAGE                                  */
                  /*                                                                   */
                  /*       //GO.INPUT1 DD DSNAME=&ASSOC6,DISP=(OLD,DELETE)             */
                  /*       //GO.OUTPUT1 DD DSNAME=ASSOC7,DISP=(NEW,PASS),              */
                  /*       //              SPACE=(CYL,(1,1)),VOLUME=SER=SCRTH2,        */
                  /*       //              DCB=(RECFM=FB,LRECL=48,BLKSIZE=7200,        */
                  /*       //              DSORG=IS,RPK=0,KEYLEN=5),UNIT=2314          */
                  /*                                                                   */
                  /*                                                                   */
                  /*   NEXT JOB STEP                                                   */
                  /*                                                                   */
                  /*   EXECUTE THESR06.                                                */
                  /*                                                                   */
```

```
        /* 'THESR05':  PROCEDURE TO WRITE ASSOCIATE WORD DATA SET AFTER SORT *                          PAGE     3

STMT LEVEL NEST

                  /*********************************************************************/
  1               THESR05:  PROCEDURE OPTIONS (MAIN);


  2     1               DECLARE
                          1 ASSOCIATE,
                            2 ASSOCSEQ  CHAR (5),         /* ORIGINAL SEQUENCE             */
                            2 ASSOCNUMC FIXED DEC (2),   /* LENGTH OF WORD                */
                            2 ASSOCSTAT FIXED DEC (1),   /* WORD STATUS                   */
                            2 ASSOCAPPR FIXED DEC (1),   /* 1 IF WORD PREVIOUSLY APP'RD */
                            2 BACKLINK  FIXED BIN(15,0),/* BACK LINK TO PRIMARY          */
                            2 ASSOCMAT  FIXED BIN(15,0),/* MATCH COUNT                   */
                            2 ASSOCFREQ FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE       */
                            2 PRIMLINK  FIXED BIN(15,0),/* SECONDARY LINK TO PRIMARY WD*/
                            2 ASSOCWD   CHAR (18);        /* ASSOCIATE WORD                */

  3     1               DECLARE
                          OUTPUT1 FILE SEQUENTIAL KEYED ENVIRONMENT (INDEXED);

  4     1               OPEN FILE(INPUT1) INPUT;
  5     1                 ON ENDFILE(INPUT1) GO TO FINISH;

  7     1               OPEN FILE(OUTPUT1) OUTPUT;

  8     1         RDRCD:
                        GET FILE(INPUT1) EDIT (ASSOCIATE)  (A(5),F(2),F(1),F(1),F(5),
                                F(5),F(5),F(5),A(18));
  9     1               WRITE FILE(OUTPUT1) FROM (ASSOCIATE) KEYFROM (ASSOCSEQ);
 10     1               GO TO RDRCD;

 11     1         FINISH:
                        CLOSE FILE(INPUT1), FILE(OUTPUT1);

 12     1               END THESR05;
 STMT LEVEL NEST
                  /* 'THESR06': PROCEDURE TO PRINT THESAUR TEXT                          */

                  /***********************************************************************/
                  /*                                                                     */
                  /*  GENERAL FLOW                                                       */
                  /*                                                                     */
                  /*  THE THESR06 PROGRAM PRINTS THOSE PRIMARY AND ASSOCIATE WORDS       */
                  /*  WHICH APPEAR IN THE TEXT.  IN ADDITION, THE PROGRAM PRINTS, AS     */
                  /*  A TREE STRUCTURE, THE RELATIONSHIP BETWEEN PRIMARY AND             */
                  /*  ASSOCIATE WORDS, I.E., THOSE ASSOCIATE WORDS WHICH ARE ALSO        */
                  /*  PRIMARY WORDS DOWN FIVE LEVELS.                                    */
                  /*                                                                     */
                  /*                                                                     */
                  /*  INPUT:                                                             */
                  /*                                                                     */
                  /*    THESPRM DATA SET;                                                */
                  /*                                                                     */
                  /*      THE USER IS REFERRED TO THE THESR01 OUTPUT1 DOCUMENTATION.     */
                  /*                                                                     */
                  /*    THESASC DATA SET;                                                */
                  /*                                                                     */
                  /*      THE USER IS REFERRED TO THE THESR05 OUTPUT1 DOCUMENTATION.     */
                  /*                                                                     */
                  /*                                                                     */
                  /*  OUTPUT:                                                            */
                  /*                                                                     */
                  /*    THE OUTPUT IS A PRINTED THESAUR OF USER SPECIFIED PRIMARY        */
                  /*    AND ASSOCIATED WORDS WHICH APPEAR IN THE TEXT.                   */
                  /*                                                                     */
                  /*                                                                     */
                  /*  SUGGESTED JOB CONTROL LANGUAGE:                                    */
                  /*                                                                     */
                  /*      //GO.INPUT1 DD DSNAME=PRIMARY,DISP=(OLD,KEEP)                  */
                  /*      //             UNIT=2314,SPACE=(CYL,(1,1)),                    */
                  /*      //             DCB=(RECFM=FB,LRECL=42,BLKSIZE=7182,DSORG=IS,   */
                  /*      //             RPF=0,KEYLEN=9),VOLUME=SER=SCRTH1               */
                  /*      //GO.INPUT2 DD DSNAME=ASSOC7,DISP=(OLD,KEEP)                   */
                  /*      //             SPACE=(CYL,(1,1)),VOLUME=SER=SCRTH2,            */
                  /*      //             SCB=(RECFM=FB,LRECL=48,BLKSIZE=7200),           */
                  /*      //             DSORG=IS,RPK=0,KEYLEN=5),UNIT=2314              */
                  /*                                                                     */
                  /*                                                                     */
                  /***********************************************************************/

                  THESR06:  PROCEDURE OPTIONS (MAIN);

                        DECLARE
                          1 PRIMARY,
                            2 PRIMNUMC  FIXED DEC (2),  /* PRIMWD LENGTH                 */
                            2 PRIMSTAT  FIXED DEC (1),  /* STATUS OF WORD                */
                            2 PRIMFREQ  FIXED BIN(15,0),/* FREQUENCY OF OCCURRENCE       */
```

```
STMT LEVEL NEST

                         2 PRMMATCNT FIXED BIN(15,0),/* MATCH COUNT                   */
                         2 FRWDLINK  FIXED BIN(15,0),/* FORWARD LINK TO ASSOC WORD    */
                         2 PRIMWD    CHAR (18);      /* PRIMARY WORD                  */

   3    1              DECLARE
                         1 ASSOCIATE,
                         2 ASSOCSEQ  CHAR (5),       /* ORIGINAL SEQUENCE             */
                         2 ASSOCNUMC FIXED DEC (2),  /* LENGTH OF WORD                */
                         2 ASSOCSTAT FIXED DEC (1),  /* 1 IF DIFFERENT FORM IN TEXT   */
                         2 ASSOCAPPR FIXED DEC (1),  /* 1 IF PREVIOUSLY APPEARED      */
                         2 BACKLINK  FIXED BIN(15,0),/* BACKWARD LINK OT PRIMARY      */
                         2 ASSOCMAT  FIXED BIN(15,0),/* MATCH COUNT                   */
                         2 ASSOCFREQ FIXED BIN(15,0),/* FREQUCNEY OF OCCURRENCE       */
                         2 PRIMLINK  FIXED BIN(15,0),/* SECONDARY LINK TO PRIMARY WD*/
                         2 ASSOCWD   CHAR (18);      /* ASSOCIATE WORD                */

   4    1              DECLARE
                         PRIMKEY       FIXED BIN(15,0) /* KEY INTO PRIMARY FILE        */
                              INITIAL (0),             /*  WITH INITIAL VALUE OF ZERO */
                         ASSOC1        FIXED BIN(15,0),/* KEY INTO FIRST LEVEL ASSOC   */
                         ASSOC2        FIXED BIN(15,0),/* END OF WORD LIST             */
                         ASSOC3        FIXED BIN(15,0),/* KEY TO SECOND LEVEL ASSOC    */
                         ASSOC4        FIXED BIN(15,0),/* END ASSOC WORD LIST          */
                         ASSOC5        FIXED BIN(15,0),/* KEY TO THIRD LEVEL ASSOC     */
                         ASSOC6        FIXED BIN(15,0),/* END ASSOC WORD LIST          */
                         ASSOC7        FIXED BIN(15,0),/* KEY TO FOURTH LEVEL ASSOC    */
                         ASSOC8        FIXED BIN(15,0),/* END ASSOC WORD LIST          */
                         KEYI          CHAR (9),       /* KEY FOR LEVEL 2              */
                         KEYJ          CHAR (9),       /* KEY FOR LEVEL 3              */
                         KEYK          CHAR (9),       /* KEY FOR LEVEL 4              */
                         KEYL          CHAR (9),       /* KEY TO LEVEL 5               */
                         SAVELINK      FIXED BIN(15,0),/* KEY TO THIRD LEVEL PRIM LIST*/
                         SAVELINK2     FIXED BIN(15,0),/* KEY TO THIRD LEVEL PRIN LIST*/
                         SAVELINK3     FIXED BIN(15,0),/* KEY TO FORTH LEVEL PRIM LIST*/
                         PAGECNTR      FIXED BIN(15,0) /* PAGE NUMBER COUNTER          */
                               INITIAL (0);

   5    1              DECLARE
                         THESPRM FIIE DIRECT KEYED ENVIRONMENT (INDEXED),
                         THESASC FILE DIRECT KEYED ENVIRONMENT (INDEXED);

   6    1              OPEN FILE(THESPRM) INPUT;
   7    1                ON ENDFILE(THESPRM) GO TO FINISH;

   9    1              OPEN FILE(THESASC) INPUT;

                                                      /*********************************/
  10    1              OPEN FILE(SYSPRINT) OUTPUT     /*                               */
                              LINESIZE (132);         /*                               */
  11    1                ON ENDPAGE(SYSPRINT) BEGIN;  /* PRINT PAGE HEADING AT BEGIN-*/
```

```
STMT LEVEL NEST

 13    2                 CALL PAGEHDG;                  /* NING OF EACH PAGE.           */
 14    2                 END;                           /*                              */
                                                        /*******************************/

 15    1              CALL PAGEHDG;                     /* INITIALIZE FIRST PAGE        */

 16    1          RDPRIM:
                      PRIMKEY = PRIMKEY + 1;
 17    1              READ FILE(THESPRM) INTO (PRIMARY) KEY (PRIMKEY);
 18    1              IF PRIMWD = (18) '9' THEN GO TO FINISH;
 20    1              PUT FILE(SYSPRINT) EDIT (PRIMFREQ,PRMMATCNT,PRIMWD,
                           '(DIFFERENT FORM APPEARS IN TEXT)')
                           (SKIP(1),COLUMN(3),F(5),COLUMN(12),F(5),COLUMN(20),A(18),
                           COLUMN(90),A((PRIMSTAT¬=0)*32));

 21    1              ASSOC1 = FRWILINK;
 22    1              READ FILE(THESPRM) INTO (PRIMARY) KEY (PRIMKEY + 1);
 23    1              ASSOC2 = FRWDLINK;

 24    1              DO I = ASSOC1 TO (ASSOC2 - 1);
 25    1    1           KEYI = I;
 26    1    1           SUBSTR(KEYI,1,5) = SUBSTR(KEYI,5,5);
 27    1    1           READ FILE(THESASC) INTO (ASSOCIATE) KEY (KEYI);
 28    1    1           IF ASSOCFREQ = 0 THEN GO TO END1;
 30    1    1           PUT FILE(SYSPRINT) EDIT (ASSOCFREQ,ASSOCMAT,(12)'-',ASSOCWD,
                           '(DIFFERENT FORM APPEARS IN TEXT)')
                           (SKIE(1),COLUMN(3),F(5),COLUMN(12),F(5),COLUMN(20),
                           A((ASSCCAPPR=0)*12),COLUMN(32),
                           A(18),COLUMN(90),A((ASSOCSTAT¬=0)*32)):
 31    1    1           IF PRIMLINK = PRIMKEY THEN GO TO END1;
 33    1    1           IF PRIMLINK -= 0 THEN DO;
 35    1    1             SAVELINK = PRIMLINK;
 36    1    1             READ FILE(THESPRM) INTO (PRIMARY) KEY (SAVELINK);
 37    1    1             ASSOC3 = FRWDLINK;
 38    1    1             READ FILE(THESPRM) INTO (PRIMARY) KEY (SAVELINK + 1);
 39    1    1             ASSOC4 = FRWDLINK;
 40    1    1             DO J = ASSCC3 TO (ASSOC4 - 1);
 41    1    2               KEYJ = J;
 42    1    2               SUBSTR(KEYJ,1,5) = SUBSTR(KEYJ,5,5);
 43    1    2               READ FILE(THESASC) INTO (ASSOCIATE) KEY (KEYJ);
 44    1    2               IF ASSCCFREQ = 0 THEN GO TO END2;
 46    1    2               PUT FILE(SYSPRINT) EDIT (ASSOCFREQ,ASSOCMAT,(12)'-',
                              ASSOCWD,'(DIFFERENT FORM APPEARS IN TEXT)')
                              (SKIP(1),COLUMN(3),F(5),COLUMN(12),F(5),COLUMN(32),
                              A((ASSCCAPPR=0)*12),COLUMN(44),
                              A(18),COLUMN(90),A((ASSOCSTAT¬=0)*32));
 47    1    2               IF PRIFLINK = SAVELINK | PRIMLINK = PRIMKEY THEN
                                  GO TO END2;
 48    1    2               IF PRIFLINK ¬= 0 THEN DO;
 49    1    2                 SAVEIINK2 = PRIMLINK;
 51    1    2
```

```
STMT LEVEL NEST

 52     1    2                   READ FILE(THESPRM) INTO (PRIMARY) KEY (SAVELINK2);
 53     1    2                   ASSOC5 = FRWDLINK;
 54     1    2                   READ FILE(THESPRM) INTC (PRIMARY) KEY (SAVELINK2 + 1);
 55     1    2                   ASSOC6 = FRWDLINK;

 56     1    2                   DO K = ASSOC5 TO (ASSOC6 - 1);
 57     1    3                     KEYK = K;
 58     1    3                     SUBSTR(KEYK,1,5) = SUBSTR(KEYK,5,5);
 59     1    3                     READ FILE(THESASC) INTO (ASSOCIATE) KEY (KEYK);
 60     1    3                     IF ASSOCFREQ = 0 THEN GO TO END3;
 62     1    3                     PUT FILE(SYSPRINT) EDIT (ASSOCFREQ,ASSOCMAT,(12)'-',
                                          ASSOCWD,'(DIFFERENT FORM APPEARS IN TEXT)')
                                          (SKIP(1),COLUMN(3),F(5),COLUMN(12),F(5),
                                          COLUMN(44),A((ASSOCAPPR=0)*12),COLUMN(56),
                                          A(18),COLUMN(90),A((ASSOCSTAT¬=0)*32));
 63     1    3                     IF PRIMLINK = PRIMKEY | PRIMLINK = SAVELINK
 64     1    3                            | PRIMLINK = SAVELINK2 THEN GO TO  END3;
 65     1    3                     IF PRIMLINK ¬= 0 THEN DO;
 67     1    3                       SAVELINK3 = PRIMLINK;
 68     1    3                       READ FILE(THESPRM) INTO (PRIMARY) KEY (SAVELINK3);
 69     1    3                       ASSOC7 = FRWDLINK;
 70     1    3                       READ FILE(THESPRM) INTO (PRIMARY) KEY (SAVELINK3 +1);
 71     1    3                       ASSOC8 = FRWDLINK;

 72     1    3                       IO L = ASSCC7 TC (ASSOC8 - 1);
 73     1    4                         KEYL = L;
 74     1    4                         SUBSTR(KEYL,1,5) = SUBSTR(KEYL,5,5);
 75     1    4                         READ FILE(THESASC) INTO (ASSOCIATE) KEY (KEYL);
 76     1    4                         IF ASSOCFREQ = 0 THEN GO TO END4;
 78     1    4                         PUT FILE(SYSPRINT) EDIT (ASSOCFREQ,ASSOCMAT,
                                          (12)'-',ASSOCWD,
                                          '(DIFFERENT FORM APEERAS IN TEXT)')
                                          (SKIP(1),COLUMN(3),F(5),COLUMN(12),F(5),
                                         COLUMN(56),A((ASSOCAPPR=0)*12),COLUMN(68),
                                          A(18),COLUMN(9C),A((ASSOCSTAT¬=0)*32));
 79     1    4       END4:               END;
 80     1    3                         END;
 81     1    3       END3:           END;
 82     1    2                     END;
 83     1    2       END2:       END;
 84     1    1                 END;
 85     1    1       END1:   END;

 86     1                  GO TO RDPRIM;

                     /************************************************************************/
                                                                /*                         */
 87     1            PAGEHDG:  PROCEDURE;                       /* SUBROUTINE TO PRINT OUT   */
                                                                /* PAGE HEADING FORMAT AND   */
```

```
STMT LEVEL NEST

                                                         /* NUMBER SUCCEEDING PAGES.    */
                                                         /*                             */
                                                         /*******************************/

 88     2               PAGECNTR = PAGECNTR + 1;
 89     2               PUT FILE(SYSPRINT) EDIT ('FREQUENCY  MATCH   PRIME WORD',
                                'PAGE #',PAGECNTR)
                                (PAGE,A,COLUMN(110),A,F(4));
 90     2               PUT FILE(SYSPRINT) EDIT ('OF OCCUR   COUNT   --LEVEL 1-  ',
                                '--LEVEL 2-  --LEVEL 3-  --LEVEL 4-  --LEVEL 5-')
                                (SKIP(1),A,A);
 91     2               PUT FILE(SYSPRINT) SKIP (1);

 92     2               RETURN;
                                                         /*******************************/
                                                         /*                             */
 93     2               END PAGEHDG;                     /* END SUBROUTINE 'PAGEHDG'.   */
                                                         /*                             */
                  /**********************************************************************/


 94     1         FINISH:
                        CLOSE FILE(THESPRM), FILE(THESASC), FILE(SYSPRINT);

 95     1               END THESR06;
```

Sample Output from List-Structure VIA

(DIFFERENT FORM...) implies a textual occurrence of the same root, but with a different ending.

```
FREQUENCY  MATCH   PRIME WORD                                                                    PAGE #   4
OF OCCUR   COUNT   --LEVEL 1-  --LEVEL 2-  --LEVEL 3-  --LEVEL 4-  --LEVEL 5-
    2       5226   ------------SAVORY
   44       5634   ------------SOUND                                           (DIFFERENT FORM APPEARS IN TEXT)
    1       6665   ------------VALID
    2       6671   ------------VALUE
    3       6738   ------------VIRTUE                                          (DIFFERENT FORM APPEARS IN TEXT)
    3       6738   ------------VIRTUOUS                                        (DIFFERENT FORM APPEARS IN TEXT)
  150       2759   GREAT                                                       (DIFFERENT FORM APPEARS IN TEXT)
    1        453   ------------ASTONISHING
    1        699   ------------BIG
    2       1237   ------------CONSEQUENTIAL
    1       1245   ------------CONSPICIOUS
   81       1517   ------------DEEP                                            (DIFFERENT FORM APPEARS IN TEXT)
    2       2004   ------------ELEVATED                                        (DIFFERENT FORM APPEARS IN TEXT)
    9       2030   ------------EMINENT                                         (DIFFERENT FORM APPEARS IN TEXT)
   43       2596   ------------FULL
  106       2721   ------------GOODLY
    7       2742   ------------GRAND
   12       2756   ------------GRAVE                                           (DIFFERENT FORM APPEARS IN TEXT)
   12       2901   ------------HEAVY                                           (DIFFERENT FORM APPEARS IN TEXT)
  228       2947   ------------HIGH                                            (DIFFERENT FORM APPEARS IN TEXT)
   49        174               ------------AIRY
    6       1810               ------------DISTINGUISHED                       (DIFFERENT FORM APPEARS IN TEXT)
    2       2004               ------------ELEVATED                            (DIFFERENT FORM APPEARS IN TEXT)
    9       2030               ------------EMINENT                             (DIFFERENT FORM APPEARS IN TEXT)
   11       2215               ------------EXCESSIVE                           (DIFFERENT FORM APPEARS IN TEXT)
    1       2269               ------------EXTRAVAGANT
  150       2759               ------------GREAT                               (DIFFERENT FORM APPEARS IN TEXT)
   13       2931               ------------HEROIC                              (DIFFERENT FORM APPEARS IN TEXT)
    2       3292               ------------INORDINATE
    4       3490               ------------KNIGHTLY
   31       3520               ------------LARGE                               (DIFFERENT FORM APPEARS IN TEXT)
    5       3642               ------------LOFTY                               (DIFFERENT FORM APPEARS IN TEXT)
    1       3709               ------------MAGNANIMOUS
   10       3726               ------------MAJESTIC                            (DIFFERENT FORM APPEARS IN TEXT)
    3       3931               ------------MONUMENTAL                          (DIFFERENT FORM APPEARS IN TEXT)
   12       4053               ------------NOBLE                               (DIFFERENT FORM APPEARS IN TEXT)
    2       4276               ------------OVERMUCH
    1       4286               ------------OVERWEENING
    1       5578               ------------SOARING
    1       5765               ------------STIFF
    9       5848               ------------SUBLIME
    5       5989               ------------TALL                                (DIFFERENT FORM APPEARS IN TEXT)
   18       6194               ------------TOWERING                            (DIFFERENT FORM APPEARS IN TEXT)
   18       6194               ------------TOWERY                              (DIFFERENT FORM APPEARS IN TEXT)
   21       3012   ------------HUGE                                            (DIFFERENT FORM APPEARS IN TEXT)
    9       3102   ------------IMMENSE
   14       3324   ------------INTENSE                                         (DIFFERENT FORM APPEARS IN TEXT)
    5       3642   ------------LOFTY                                           (DIFFERENT FORM APPEARS IN TEXT)
   10       3726   ------------MAJESTIC                                        (DIFFERENT FORM APPEARS IN TEXT)
   36       3857   ------------MIGHTY                                          (DIFFERENT FORM APPEARS IN TEXT)
    3       3931   ------------MONUMENTAL                                      (DIFFERENT FORM APPEARS IN TEXT)
   76       3965   ------------MUCH
   12       4053   ------------NOBLE                                           (DIFFERENT FORM APPEARS IN TEXT)
    9       4517   ------------POINTED                                         (DIFFERENT FORM APPEARS IN TEXT)
    6       4576   ------------PRECIOUS
    5       4627   ------------PRODIGIOUS
    1       4942   ------------REMARKABLE
    9       5848   ------------SUBLIME
```

APPENDIX E

MAPTEXT Program Listing and Output

by

Barbara Snider

```
STMT LEVEL NEST
  1              MAPTEXT: PROCEDURE(PARM) CPTICNS(MAIN);
                                                                                                  MPT00020
                                                                                                  MPT00030
                                                                                                  MPT00050
                 /*******************************************************************/  MPT00070
                 /*                                                                 */  MPT00080
                 /*   MAPTEXT IS A PROGRAM, WRITTEN IN PL/1, THAT TAKES VERBAL TEXT */  MPT00081
                 /*   AND PRODUCES A NCN-VERBAL REPRESENTATION OF THE ENTIRE TEXT OR */  MPT00082
                 /*   SELECTED PORTICNS THEREOF.   MAPTEXT IS A TYPE OF STYLISTIC    */  MPT00083
                 /*   ANALYSIS THAT LETS YOU ICCK AT PATTERNS IN WRITINGS WITHOUT    */  MPT00084
                 /*   LOCKING AT WORIS THEMSELVES.                                   */  MPT00085
                 /*                                                                 */  MPT00086
                 /*                                                                 */  MPT00087
                 /*   THE INPUT TO MAPTEXT IS THE OUTPUT FRCM 'INDEX', WHICH PRODUCES*/  MPT00088
                 /*   A 36-CHARACTER RECORD FOR EACH WCRD IN THE TEXT.(FOR MORE      */  MPT00089
                 /*   INFORMATION REFER TO THE PROGRAM DOCUMENTATION FOR 'INDEX'.)   */  MPT00090
                 /*                                                                 */  MPT00091
                 /*                                                                 */  MPT00092
                 /*   EACH RECORD IS READ INTO THE STRUCTURE TEXT, WHOSE FORMAT IS:  */  MPT00093
                 /*             NUMCHAR - THE NUMBER OF CHARACTERS IN THE WORD (2)   */  MPT00094
                 /*             TEXTINDX - THE VOLUME (2), CHAPTER (3),              */  MPT00095
                 /*                              PARAGRAPH (3), SENTENCE (5) OF WORD */  MPT00096
                 /*             WDNSENT - NUMBER WORD IN SENTENCE (3)                */  MPT00097
                 /*             TEXTWORD - THE WORD ITSELF, FROM TEXT (18)           */  MPT00098
                 /*                                                                 */  MPT00099
                 /*                                                                 */  MPT00100
                 /*                                                                 */  MPT00160
                 /*                                                                 */  MPT00170
                 /*   THE TEXT IS READ IN  ONE WORD AT A TIME. IF THE WORD APPEARS   */  MPT00180
                 /*   IN THE TABLE, THE SYMBOL ASSOCIATED WITH THAT WORD IS PRINTED. */  MPT00190
                 /*   IF THERE IS NO MATCH, BUT THE FIRST 3 CHARACTERS OF 2 WORDS    */  MPT00200
                 /*   MATCH, THE SUBROUTINE STEM IS CALLED.                          */  MPT00210
                 /*   IF NC MATCH OCCURS, THE 'NO MATCH' SYMBOL ('.') IS PRINTED.    */  MPT00220
                 /*                                                                 */  MPT00230
                 /*     TBLCRST:  ROUTINE TO BUILD TABLE DENOTING INDEXED RECORDS    */  MPT00280
                 /*   TO BE PROCESSEI.  FOR FURTHER INFCRMATION ON THIS ROUTINE      */  MPT00290
                 /*   REFER TO PROGRAM  DOCUMENTATION OF TBLCRST. THE USER  MUST     */  MPT00300
                 /*   SPECIFY THE LEVEL OF PROCESSING BY:                            */  MPT00310
                 /*                                                                 */  MPT00320
                 /*     (1) PASSING THE PROCESSING TYPE CODE AS 'PROCTYP'.           */  MPT00330
                 /*         THIS   PARAMETER IS PASSEI BY THE JCL:                   */  MPT00340
                 /*                                                                 */  MPT00350
                 /*                                                                 */  MPT00360
                 /*         //  EXEC  PL1,PARM.PL1='X',PARM.GO='*,Y'                 */  MPT00370
                 /*                                                                 */  MPT00380
                 /*         WHERE '*' INDICATES LEVEL OF PROCESSING AND MAY BE EQUAL:*/  MPT00390
                 /*                                                                 */  MPT00400
                 /*             A    PROCESS ALL DATA                                */  MPT00410
                 /*             V    PROCESS ON THE VOLUME LEVEL                     */  MPT00420
                 /*             C    PROCESS ON THE CHAPTER LEVEL                    */  MPT00430
                 /*             P    PROCESS ON THE PARAGRAPH LEVEL                  */  MPT00440
                 /*             S    PRCCESS ON THE SENTENCE LEVEL                   */  MPT00450
```

```
STMT LEVEL NEST

                 /*     (2) SUPPLYING THE INDEX INFORMATION FOR DATA TO BE           */  MPT00460
                 /*         PROCESSED, WHICH IMMEDIATELY FOLLOWS THE CARD:           */  MPT00470
                 /*                                                                  */  MPT00480
                 /*         //GO.SYSIN  DD  *                                        */  MPT00490
                 /*                                                                  */  MPT00500
                 /*         (IF THE LEVEL OF PROCESSING IS 'A', THERE WILL BE        */  MPT00510
                 /*         NO INDEXING INFORMATION, SINCE ALL RECORDS ARE PROCESSED)*/  MPT00520
                 /*                                                                  */  MPT00521
                 /*                                                                  */  MPT00522
                 /*      IF  THE LEVEL OF PROCESSING IS 'C' AND YOU WOULD LIKE TO    */  MPT00530
                 /*      PROCESS VOLUME 1, CHAPTER 3,     AND                        */  MPT00540
                 /*              VOLUME 1, CHAPTERS 6-10;                            */  MPT00550
                 /*          (1,3),((1,6),(1,10))                                    */  MPT00560
                 /*                                                                  */  MPT00570
                 /*      IF  THE LEVEL OF PROCESSING IS 'P' AND YOU WOULD LIKE TO    */  MPT00580
                 /*      PROCESS VOLUME2, CHAPTER 3,  PARAGRAPHS 7-12 , AND          */  MPT00590
                 /*              VOLUME 3, CHAPTER 4,  PARAGRAPH 6, AND              */  MPT00600
                 /*              VOLUME 7, CHAPTER 2,  PARAGRAPHS 3-38:              */  MPT00610
                 /*          ((2,3,7),(2,3,12)),(3,4,6),                        *    */  MPT00620
                 /*          ((7,2,3),(7,2,38))                                      */  MPT00630
                 /*                                                                  */  MPT00640
                 /*       NOTE: AN ASTERISK IN COLUMN 72 INDICATES A CONTINUATION    */  MPT00660
                 /*       CARD FOLLOWS. INDEX MATERIAL MUST BEGIN IN COLUMN 1.       */  MPT00670
                 /*                                                                  */  MPT00675
                 /*       FOR FURTHER INFORMATION REFER TO THE SECTION ON TBLCRST.   */  MPT00680
                 /*                                                                  */  MPT00682
                 /*       THE SPECIAL WORDS TO BE SEARCHED FOR AND THEIR SYMBOLS ARE */  MPT00684
                 /*       PUNCHED ON CARDS, ONE WORD PER CARD:  THE SYMBOL IN        */  MPT00686
                 /*       COLUMN 1, (COLUMNS 2 - 4 BLANK) AND THE WORD BEGINNING IN  */  MPT00688
                 /*       COLUMN 5 (A MAXIMUM OF 18 CHARACTERS PER WORD).            */  MPT00690
                 /*                                                                  */  MPT00692
                 /*       THESE CARDS MAY BE IN ANY ORDER, AND IMMEDIATELY  FOLLOW   */  MPT00694
                 /*       THE INDEX INFORMATION ( UNLESS THE LEVEL OF PROCESSING     */  MPT00696
                 /*       IS 'A', IN WHICH CASE THEY FOLLOW THE //GO.SYSIN DD * )    */  MPT00698
                 /*                                                                  */  MPT00700
                 /*       FOR EXAMPLE, TO PROCESS ONLY THE FIRST CHAPTER OF VOLUMN 1 */  MPT00702
                 /*      //GO.SYSIN   DD   *                                         */  MPT00704
                 /*       (1,1)                                                      */  MPT00706
                 /*      W   WAR                                                     */  MPT00707
                 /*      P   PEACE                                                   */  MPT00708
                 /*                                                                  */  MPT00709
                 /*                                                                  */  MPT00710
                 /*     STEM:  ROUTINE THAT CHECKS TO SEE IF 2 WORDS HAVE THE SAME   */  MPT00715
                 /*     ROOT.  IF THEY DO, SAME_ROOT IS SET EQUAL TO 1, WHICH TELLS  */  MPT00720
                 /*     THE MAIN PROGRAM THAT A MATCH HAS OCCURED.  IF SAME_ROOT     */  MPT00730
                 /*     IS SET EQUAL TO 0, THERE WAS  NO MATCH.                      */  MPT00740
                 /*                                                                  */  MPT00752
                 /*     THE USER MUST SPECIFY ON THE 'EXEC' CARD WHETHER OR NOT HE   */  MPT00754
                 /*     WISHES TO EXEC THE STEM ROUTINE:                             */  MPT00756
                 /*                                                                  */  MPT00758
```

```
        MAPTEXT: PROCEDURE(PARM) OPTIONS(MAIN);                                                           PAGE       4

STMT LEVEL NEST

                  /*      //  EXEC  PL1,PARM.PL1='X',PARM.GO='*,$'                                */  MPT00760
                  /*                                                                              */  MPT00762
                  /*    WHERE '$' INDICATES THE STEM OPTION (STEM_OP) AND MAY BE  :               */  MPT00764
                  /*            Y    YES, EXECUTE STEM                                            */  MPT00765
                  /*            N    NO, DO NOT EXECUTE STEM                                      */  MPT00766
                  /*                                                                              */  MPT00767
                  /*******************************************************************************/  MPT00780
                                                                                                      MPT00800
  2     1                 DCL PARM    CHAR(40)  VARYING;  /*PROCESS LEVEL PARAMETER   */ MPT00810
  3     1                 DCL INPUT FILE RECORD INPUT,                                   MPT00820
                             1 TEXT,                      /* INPUT RECORD              */ MPT00830
                                2  NUMCHAR    CHAR(2),    /*NO. CHAR. IN INPUT WORD    */ MPT00840
                                2  TEXTINDX   CHAR(13),   /* VOL,CHAPT,PARAG,SENTENCE */ MPT00850
                                2  WINSENT   CHAR(3),     /* NO. OF WORD IN SENTENCE  */ MPT00860
                                2  TEXTWORD   CHAR(18);   /* WORD                      */ MPT00870
  4     1                 DCL    1 OLDTEXT,               /* LAST RECORD NO. PROCESSED*/ MPT00872
                                 2  CLDV    CHAR(2) ,     /* VOLUME                    */ MPT00874
                                 2  CLDC    CHAR(3) ,     /* CHAPTER                   */ MPT00876
                                 2  CLDP    CHAR(3) ,     /* PARAGRAPH                 */ MPT00878
                                 2  CLDS    CHAR(5) ;     /* SENTENCE                  */ MPT00879
  5     1                 DCL   OLD_TEXT      CHAR(13) DEFINED OLDTEXT;                  MPT00881
                                                                                         MPT00880
  6     1                 DCL    1  WORK,                 /*  WORKAREA FOR SORT        */ MPT00890
                                 2  WORKA    CHAR(1),                                    MPT00900
                                 2  WORKB    FIXED DEC(2),                               MPT00910
                                 2  WORKC    CHAR(18) VARYING;                           MPT00920
                                                                                         MPT00925
  7     1                 DCL    1  RTABLE(200),          /* SPECIAL LINGUISTIC UNITS */ MPT00930
                                 2  SYMBOL   CHAR(1),     /*SYMBOL ASSOC. WITH UNIT    */ MPT00940
                                 2  IGTH     FIXED DEC(2),/*LENGTH OF EACH UNIT        */ MPT00950
                                 2  RTABWORD CHAR(18) VARYING; /* SPECIAL WORDS        */ MPT00960
                                                                                         MPT00970
  8     1                DCL    BYPASS        LABEL;                                     MPT00980
  9     1               DECLARE                                                          MPT00990
                                BLK           CHAR(1),    /* DUMMY CHARACTER           */ MPT01000
                                CARC          CHAR(1),    /*USED IN A TO FIND LGTH     */ MPT01010
                                PGECNTR       FIXED DEC(3) INIT(0),  /* PAGE COUNTER   */ MPT01015
                                IDX           FIXED BIN(15,0) INITIAL(0),                MPT01020
                                NUMENTRY      FIXED DEC(3) INITIAL(200), /*NO. IN TBL */ MPT01040
                                PROCTYP       CHAR(1),    /*LEVEL OF PROCESSING        */ MPT01050
                                LNG           FIXED DEC(2), /* USED TO FIND LGTH       */ MPT01060
                                OLDSENT       CHAR(5)  INITIAL(0),  /*LAST SENT (NO.) */ MPT01070
                                SAME_ROOT     FIXED DEC(1), /* SET BY STEM             */ MPT01090
                                STEM_OP       CHAR(1),    /* STEM ROUTINE OPTION       */ MPT01095
                                STRLGTH       FIXED DEC(2),  /*LENGTH OF INDEX STRING */ MPT01100
                                TBL(200)      CHAR(13),   /*TBL OF RECORDS TO BE PROC.*/ MPT01110
                                TAB           FIXED BIN(15,0) INITIAL(0),                MPT01120
                                WDLGTH        FIXED DEC(2), /*DEC. REP. OF NUMCHAR     */ MPT01130
                                WORD          CHAR(18) VARYING, /*USED TO FIND LGTH    */ MPT01140
                                SYSIN FILE;               /* EXPLICIT DECLARATION      */ MPT01150
```

```
          MAPTEXT: PROCEDURE(PARM) OPTIONS(MAIN);                                                           PAGE    5

 STMT LEVEL NEST

                                                                                                   MPT01160
                      /**********************************************************************/ MPT01170
                      /*                                                                    */ MPT01172
                      /*     GET (FROM JCL) THE PROCTYP AND STEM OPTION PARAMETERS          */ MPT01173
                      /*                                                                    */ MPT01174
                      /**********************************************************************/ MPT01175
                                                                                               MPT01230
  10      1                  PRCCTYP=SUBSTR(PARM,1,1);                                         MPT01231
  11      1                  STEM_OP = SUBSTR(PARM,3,1);                                       MPT01232
                                                                                               MPT01235
                      /*********************************************************************/  MPT01237
                      /*                                                                   */  MPT01239
                      /*    IF STEM_OP EQUAL 'Y', THEN SET BYPASS EQUAL LAB2, AND STEM     */  MPT01242
                      /*    WILL BE EXECUTED.                                              */  MPT01243
                      /*                                                                   */  MPT01244
                      /*    IF STEM_OP EQUAL 'N', THEN SET BYPASS EQUAL ENDEE, AND STEM    */  MPT01245
                      /*    WILL NOT BE EXECUTED.                                          */  MPT01246
                      /*                                                                   */  MPT01247
                      /*********************************************************************/  MPT01248
                                                                                               MPT01249
  12      1                  IF STEM_OP = 'Y'  THEN DO;                                        MPT01250
  14      1                  BYPASS = LAB2;                                                    MPT01251
  15      1                  END;                                                              MPT01252
  16      1                  ELSE DO;                                                          MPT01253
  17      1                  BYPASS = ENDEE;                                                   MPT01254
  18      1                  END;                                                              MPT01255
                                                                                               MPT01260
  19      1                  OPEN FILE(SYSIN) INPUT;                                           MPT01270
  20      1                  CPEN FILE(SYSPRINT) OUTPUT LINESIZE(132);                         MPT01280
                                                                                               MPT01290
                      /*********************************************************************/  MPT01291
                      /*                                                                   */  MPT01292
                      /* PROCEDURE TO PUT A PAGE NUMBER AT THE TOP OF EACH PAGE.           */  MPT01293
                      /*                                                                   */  MPT01294
                      /*********************************************************************/  MPT01295
  21      1                  ON ENDPAGE(SYSPRINT) BEGIN;                                       MPT01296
  23      2                  CALL PAGECNTR;                                                    MPT01297
  24      2                  END;                                                              MPT01298
  25      1                  PAGECNTR: PROCEDURE;                                              MPT01299
  26      2                  PGECNTR = PGECNTR + 1;                                            MPT01300
  27      2                  PUT FILE(SYSPRINT) EDIT ('PAGE',PGECNTR)                          MPT01310
                                    (PAGE,COLUMN(120),A,X(1),F(3));                            MPT01315
  28      2                  PUT FILE(SYSPRINT) SKIP;                                          MPT01320
  29      2                  END;                                                              MPT01333
                                                                                               MPT01334
                      /*********************************************************************/  MPT01335
                      /*                                                                   */  MPT01336
                      /*  CALL TBLCRST TO DETERMINE RECORDS TO BE PROCESSED                */  MPT01337
                      /*                                                                   */  MPT01338
                      /*********************************************************************/  MPT01339
```

```
STMT LEVEL NEST
                                                                                          MPT01370
                                                                                          MPT01380
 30     1              CALL TBLCRST (TBL,NUMENTRY,PROCTYP);                               MPT01390
                                                                                          MPT01400
 31     1                CALL PAGECNTR;                                                   MPT01405
 32     1                PUT FILE(SYSPRINT) EDIT ('TABLE OF INDEX INFORMATION FOR',       MPT01410
                               ' RECORDS TO BE PROCESSED.','LEVEL OF PROCESSING ',        MPT01420
                               'SPECIFIED IS (',PROCTYP,').','  NUMBER OF TABLE ',        MPT01430
                               'ENTRIES EQUALS (',NUMENTRY,').',' V  C  P  S')            MPT01440
                               (SKIP(0),A,A,SKIP(2),COLUMN(5),A,A,A,A,A,A,F(3),A,         MPT01445
                               SKIP(2),COLUMN(10),A);                                     MPT01460
 33     1                PUT FILE(SYSPRINT) SKIP(2);                                      MPT01470
                                                                                          MPT01480
 34     1                DO I = 1 TO NUMENTRY;                                            MPT01490
 35     1     1          PUT FILE(SYSPRINT) EDIT  (TBL(I))  (SKIP(1),COLUMN(10),A);       MPT01500
 36     1     1          END;                                                             MPT01510
                                                                                          MPT01520
 37     1                ON ENDFILE (SYSIN) GO TO LOOPB;                                  MPT01530
                                                                                          MPT01540
 39     1                PUT FILE(SYSPRINT) PAGE;                                         MPT01550
                                                                                          MPT01560
           /*******************************************************************/ MPT01565
           /*                                                                 */ MPT01566
           /*  CREATE TABLE OF WORDS TO CHECK FOR(AND THEIR SYMBOL)           */ MPT01596
           /*  AND DETERMINE LENGTH OF EACH WORD.                             */ MPT01606
           /*                                                                 */ MPT01607
           /*******************************************************************/ MPT01609
                                                                                          MPT01610
 40     1             A:   IDX=IDX+1;                                                     MPT01630
 41     1                GET FILE(SYSIN) EDIT (SYMBOL(IDX), RTABWORD(IDX),BLK)            MPT01640
                                              (A(1), X(3), A(18), X(57), A(1));           MPT01650
 42     1                LNG = 0;                                                         MPT01660
 43     1                WORD = '';                                                       MPT01670
 44     1         LOOPA: LNG = LNG + 1;                                                   MPT01680
 45     1                CARC=(SUBSTR(RTABWORD(IDX),LNG,1));                              MPT01690
 46     1                IF CARC =' '     THEN GO TO RTN;                                 MPT01700
 48     1                WORD = WORD||CARC;                                               MPT01720
 49     1                GO TO LOOPA;                                                     MPT01730
                                                                                          MPT01740
 50     1           RTN: RTABWORD(IDX) = WORD;                                            MPT01750
 51     1                LGTH(IDX)  = LNG - 1;                                            MPT01760
 52     1                GO TO A;                                                         MPT01770
                                                                                          MPT01780
                                                                                          MPT01790
                                                                                          MPT01810
           /*******************************************************************/ MPT01820
           /*                                                                 */ MPT01830
           /*  SORT THE TABLE ALPHABETICALLY AND PRINT IT.                    */ MPT01840
           /*                                                                 */ MPT01850
           /*******************************************************************/ MPT01860
```

```
STMT LEVEL NEST

                                                                                              MPT01861
                                                                                              MPT01862
                                                                                              MPT01870
 53     1        LOOPB:  DO L = 1 TO (IDX-1);                                                 MPT01880
 54     1    1           IF RTABWORD(L)>RTABWORD(L+1)    THEN DO;                             MPT01890
 56     1    1              DO M=L+1 TO 2 BY -1 WHILE (RTABWORD(M)<RTABWORD(M-1));            MPT01900
 57     1    2              WORK=RTABLE(M);                                                   MPT01910
 58     1    2              RTABLE(M)=RTABLE(M-1);                                            MPT01920
 59     1    2              RTABLE(M-1)=WORK;                                                 MPT01930
 60     1    2              END;                                                              MPT01940
 61     1    1              END;                                                              MPT01950
 62     1    1           END;                                                                 MPT01960
 63     1                CALL PAGECNTR;                                                       MPT01961
 64     1                PUT FILE(SYSPRINT)  EDIT ('TABLE OF LINGUISTIC SUBSTITUTES,',        MPT01962
                               'SORTED ALPHABETICALLY','SYMBOL   ','WORD')                    MPT01963
                               (SKIP(0),A,A,SKIP(2),A,A);                                     MPT01964
                                                                                              MPT01970
 65     1                DO L = 1 TO (IDX);                                                   MPT01980
 66     1    1           PUT FILE(SYSPRINT) EDIT (SYMBOL(L), RTABWORD(L))                     MPT01990
                                (SKIP, A(1), X(7), A(18));                                    MPT02000
 67     1    1           END;                                                                 MPT02010
                                                                                              MPT02020
         /*********************************************************************************/ MPT02021
         /*                                                                               */ MPT02022
         /*  SORT THE TABLE OF SPECIAL LINGUISTIC UNITS BY SYMBOL, AND PRINT              */ MPT02050
         /*                                                                               */ MPT02051
         /*********************************************************************************/ MPT02052
                                                                                              MPT02080
 68     1                DO L = 1 TO (IDX-1);                                                 MPT02090
 69     1    1           IF SYMBOL(L)>SYMBOL(L+1)    THEN DO;                                 MPT02100
 71     1    1              DO M=L+1 TO 2 BY -1  WHILE (SYMBOL(M)<SYMBOL(M-1));               MPT02110
 72     1    2              WORK=RTABLE(M);                                                   MPT02120
 73     1    2              RTABLE(M)=RTABLE(M-1);                                            MPT02130
 74     1    2              RTABLE(M-1)=WORK;                                                 MPT02140
 75     1    2              END;                                                              MPT02150
 76     1    1              END;                                                              MPT02160
 77     1    1           END;                                                                 MPT02170
                                                                                              MPT02180
 78     1                CALL PAGECNTR;                                                       MPT02181
 79     1                PUT FILE(SYSPRINT)  EDIT ('TABLE OF LINGUISTIC SUBSTITUTES,',        MPT02182
                               'SORTED BY SYMBOL','SYMBOL   ','WORD')                         MPT02183
                               (SKIP(0),A,A,SKIP(2),A,A);                                     MPT02184
                                                                                              MPT02185
 80     1                DO L=1 TO (IDX);                                                     MPT02200
 81     1    1           PUT FILE(SYSPRINT) EDIT (SYMBOL(L), RTABWORD(L))                     MPT02215
                                (SKIP,A(1), X(7), A(18));                                     MPT02216
 82     1    1           END;                                                                 MPT02230
                                                                                              MPT02240
                                                                                              MPT02250
 83     1                IF PROCTYP = 'V' THEN DO;                                            MPT02260
```

```
          MAPTEXT: PROCEDURE(PARM) OPTIONS(MAIN);

 STMT LEVEL NEST

  85      1                 STRLGTH = 2;                                                   MPT02270
  86      1                 GO TO ARND1;                                                   MPT02280
  87      1                 END;                                                           MPT02290
  88      1               IF PROCTYP = 'C' THEN DO;                                        MPT02300
  90      1                 STRLGTH = 5;                                                   MPT02310
  91      1                 GO TO ARND1;                                                   MPT02320
  92      1                 END;                                                           MPT02330
  93      1               IF PROCTYP = 'P' THEN DO;                                        MPT02340
  95      1                 STRLGTH = 8;                                                   MPT02350
  96      1                 GO TO ARND1;                                                   MPT02360
  97      1                 END;                                                           MPT02370
  98      1               IF PROCTYP = 'S' THEN DO;                                        MPT02380
 100      1                 STRLGTH = 13;                                                  MPT02390
 101      1                 END;                                                           MPT02410
                                                                                           MPT02420
             /*********************************************************************/ MPT02431
             /*                                                                   */ MPT02432
             /* BEGIN PROCESSING OF TEXT.  TEXT MUST BE IN SEQUENCE ON INDEX DATA*/ MPT02433
             /*      IN RECORD (PCSITIONS 3 THROUGH 18).                         */ MPT02435
             /*                                                                   */ MPT02436
             /*********************************************************************/ MPT02437
                                                                                           MPT02500
 102      1  ARND1:                                                                        MPT02510
                       OPEN FILE(INPUT) INPUT;                                             MPT02520
 103      1            ON ENDFILE (INPUT) GO TO FINISH;                                    MPT02530
 105      1            CALL PAGECNTR;                                                      MPT02535
 106      1            PUT FILE(SYSPRINT) EDIT ('IN THE FCLLOWING PRINTOUT, ALL ',         MPT02550
                              'INDEXED TEXT WORDS HAVE BEEN REPLACED BY A (.).',          MPT02560
                              'WHEN THE INDEXED TEXT WORD OR A SUITABLE FORM OF THE ',    MPT02570
                              'WORD HAS BEEN SPECIFIED BY THE USER AS A WORD OF ',        MPT02580
                              'INTEREST, THE SPECIAL SYMBCL ASSOCIATED WITH THE ',        MPT02590
                              'WORD IS SUBSTITUTED INSTEAD OF THE  (.).',                 MPT02600
                              'SENTENCES ARE TERMINATED WITH A (/).',                     MPT02610
                              'V=VCLUME, C=CHAPTER, P=PARAGRAGH, S=SENTENCE,',            MPT02615
                              ' OF THE LAST RECORD PROCESSED ON EACH PRINT-LINE.',        MPT02617
                              ' V    C    P      S')                                      MPT02620
                              (SKIP(0),A,A,SKIP(1),COLUMN(10),A,A,SKIP(1),COLUMN(10),     MPT02621
                              A,A,SKIP(2),COLUMN(10),A,SKIP(1),COLUMN(10),A,A,            MPT02622
                              SKIP(2),COLUMN(110),A);                                     MPT02623
 107      1            PUT FILE(SYSPRINT) SKIP(2);                                         MPT02650
                                                                                           MPT02660
                                                                                           MPT02670
 108      1            I=1;                                                                MPT02680
 109      1            J=1;                                                                MPT02690
                                                                                           MPT02700
 110      1            READ FILE (INPUT) INTO (TEXT);                                      MPT02710
 111      1  FIRST:    CLDSENT = SUBSTR(TEXTINDX,9,5);                                     MPT02720
 112      1            IF PROCTYP = 'A' THEN GO TO DD;                                     MPT02730
 114      1            GO TO LAB1;                                                         MPT02740
                                                                                           MPT02750
          MAPTEXT: PROCEDURE(PARM) OPTIONS(MAIN);

 STMT LEVEL NEST

                    /*******************************************************************/ MPT02751
                    /*                                                                 */ MPT02752
                    /*  READ ANOTHER RECORD.                                           */ MPT02753
                    /*                                                                 */ MPT02754
                    /*******************************************************************/ MPT02755
                                                                                         MPT02756
                                                                                         MPT02757
  115      1         BB:     READ FILE (INPUT) INTO (TEXT);                               MPT02810
  116      1                 IF PRCCTYP = 'A' THEN GO TO DD;                              MPT02820
                                                                                         MPT02840
                    /*******************************************************************/ MPT02841
                    /*                                                                 */ MPT02842
                    /*  CHECK TO SEE IF THIS RECORD IS TO BE PROCESSED                 */ MPT02844
                    /*                                                                 */ MPT02845
                    /*******************************************************************/ MPT02846
                                                                                         MPT02900
  118      1        IAB1:    IF SUBSTR(TEXTINDX,1,STRLGTH) = SUBSTR(TBL(I),1,STRLGTH)    MPT02910
  119      1                       THEN GO TO DD;                                         MPT02920
  120      1                 IF SUBSTR(TEXTINDX,1,STRLGTH) > SUBSTR(TBL(I),1,STRLGTH)    MPT02930
  121      1                       THEN DO;                                               MPT02940
  122      1                   I = I + 1;                                                 MPT02950
  123      1                   GO TO LAB1;                                                MPT02960
  124      1                 END;                                                         MPT02970
  125      1                 GO TO BB;                                                    MPT02990
                                                                                         MPT03000
  126      1         DD:     IF SUBSTR(TEXTINDX,9,5) = CLDSENT                            MPT03010
  127      1                 THEN GO TO PRNT;                                             MPT03015
                                                                                         MPT03030
                    /*******************************************************************/ MPT03035
                    /*                                                                 */ MPT03036
                    /* WHEN YOU COME TO THE END OF SENTENCE, PRINT A '/' AND SKIP LINE. */ MPT03037
                    /*                                                                 */ MPT03038
                    /*******************************************************************/ MPT03039
                                                                                         MPT03090
  128      1                 PUT FILE(SYSPRINT) EDIT ('/ ') (A(2));                       MPT03091
                                                                                         MPT03110
  129      1                 J = J + 2;                                                   MPT03111
  130      1                 IF (OLDSENT + 1) ¬= SUBSTR(TEXTINDX,9,5)                     MPT03112
  131      1                 THEN DO;                                                     MPT03113
  132      1                 OLDSENT=SUESTR(TEXTINDX,9,5);                                MPT03114
  133      1                 PUT FILE(SYSPRINT) EDIT (OIDV,OIDC,OLDP,OLDS)                MPT03115
                                   (COLUMN(110),A(2), X(2),A(3),X(2),A(3),X(2),A(5));    MPT03116
  134      1                 PUT FILE(SYSPRINT) SKIP(2);                                  MPT03117
  135      1                 J = 1;                                                       MPT03118
  136      1                 GO TO EE;                                                    MPT03119
  137      1                 END;                                                         MPT03120
  138      1                 OLDSENT = SUBSTR(TEXTINDX,9,5);                              MPT03125
  139      1        PRNT:    IF J < 100    THEN GO TO EE;                                 MPT03150
                                                                                         MPT03170
                                                                                         MPT03176
```

```
STMT LEVEL NEST

                  /*******************************************************************/ MPT03177
                  /*                                                                 */ MPT03178
                  /*  WHEN YOU COME TO THE END OF THE PRINT_LINE, SKIP               */ MPT03179
                  /*  TO COLUMN 110 AND PRINT THE VOLUME,CHAPTER, PARAGRAPH,         */ MPT03180
                  /*   SENTENCE OF THE LAST RECORD PROCESSED. BEGIN A NEW LINE       */ MPT03181
                  /*                                                                .*/ MPT03182
                  /*******************************************************************/ MPT03183
                                                                                        MPT03250
 141      1              PUT FILE(SYSPRINT) EDIT (OLDV,OLDC,OLDP,OLDS)                  MPT03251
                               (COLUMN(110),A(2), X(2),A(3),X(2),A(3),X(2),A(5));       MPT03254
 142      1              PUT FILE(SYSPRINT) SKIP;                                       MPT03262
 143      1              J = 1;                                                         MPT03280
                                                                                        MPT03290
                  /*******************************************************************/ MPT03291
                  /*                                                                 */ MPT03292
                  /*  COMPARE EACH RECORD WITH EACH WORD IN THE TABLE.               */ MPT03293
                  /*                                                                 */ MPT03294
                  /*  IF A MATCH OCCURS, PRINT THE SYMBOL(CHARACTER)                 */ MPT03295
                  /   ASSOCIATED WITH THAT WORD                                      */ MPT03296
                  /   IF NO MATCH OCCURS, PRINT THE 'NC MATCH' SYMBOL                */ MPT03297
                  /*  ('.').                                                         */ MPT03298
                  /*                                                                 */ MPT03299
                  /********************************************* ********************/ MPT03300
                                                                                        MPT03301
 144      1        EE:    WDLGTH=DECIMAL(NUMCHAR);                                      MPT03420
 145      1               DO K=1 TO (IDX);                                              MPT03430
 146      1     1         IF TEXTWORD=RTABWORD(K)     THEN GO TO MATCH;                 MPT03440
 148      1     1         GO TO BYPASS;                                                 MPT03441
 149      1     1  LAB2:  IF SUBSTR(TEXTWORD,1,3)¬=SUBSTR(RTABWORD(K),1,3)              MPT03450
 150      1     1         THEN GO TO ENDEE;                                             MPT03460
                                                                                        MPT03461
                  /*******************************************************************/ MPT03462
                  /*                                                                 */ MPT03463
                  /*  CALL THE SUBROUTINE STEM (EXPLAINED ABOVE)                     */ MPT03464
                  /*                                                                 */ MPT03465
                  /*******************************************************************/ MPT03466
                                                                                        MPT03467
 151      1     1         CALL STEM(SAME_ROOT,WDLGTH,TEXTWORD,LGTH(K),RTABWORD(K));     MPT03520
 152      1     1         IF SAME_ROOT = 0 THEN GO TO ENDEE;                            MPT03530
                                                                                        MPT03531
 154      1     1  MATCH: PUT FILE(SYSPRINT) EDIT (SYMBOL(K))  (A(1));                  MPT03532
 155      1     1         GO TO GG;                                                     MPT03550
 156      1     1  ENDEE: END;                                                          MPT03560
                                                                                        MPT03570
 157      1               PUT FILE(SYSPRINT) EDIT ('.') (A(1));                         MPT03571
 158      1          GG:  J=J + 1;                                                      MPT03590
 159      1               CLD_TEXT=TEXTINDX;                                            MPT03600
 160      1               GO TO BB;                                                     MPT03610
                                                                                        MPT03615
 161      1        FINISH: PUT FILE(SYSPRINT)  EDIT ('/ ')  (A(2));                     MPT03620
```

```
STMT LEVEL NEST

162      1                IF (J¬=1 & J<100)          TPEN                                   MPT03621
163      1                PUT FILE(SYSPRINT)  EDIT (OLDV,OLDC,CLDP,OLDS)                    MPT03622
                                (COLUMN(110),A(2), X(2),A(3),X(2),A(3),X(2),A(5));          MPT03623
164      1                CLOSE FILE(SYSIN),                                                MPT03630
                                FILE(SYSPRINT)                                              MPT03631
                                FILE(INPUT);                                                MPT03632
                                                                                            MPT03650
165      1                END MAPTEXT;                                                      MPT03660
```

Sample Output from MAPTEXT

(A represents those words having the root ARM, and F those words having the root FORCE.)

```
                                                                                                              PAGE    6
IN THE FOLLOWING PRINTOUT, ALL INDEXED TEXT WORDS HAVE BEEN REPLACED BY A (.).
          WHEN THE INDEXED TEXT WORD OR A SUITABLE FORM OF THE WORD HAS EEEN SPECIFIED BY THE USER AS A WORD OF
          INTEREST, THE SPECIAL SYMBOL ASSOCIATED WITH THE WORD IS SUBSTITUTED INSTEAD OF THE  (.).

          SENTENCES ARE TERMINATED WITH A (/).
          V=VOLUME, C=CHAPTER, P=PARAGRAGH, S=SENTENCE, OF THE LAST RECORI PROCESSED ON EACH PRINT-LINE.
          (IN THE CASE OF POETRY, S=LINE.
           IN THE CASE OF PLAYS, C=ACT, P=SCENE.
           IN THE CASE OF SPEECHES, V=SERIES, C=SESSION,  P=SPEAKER.)

                                                                                                      V     C     P      S
....MS.....D../ ...................................../ .M..............M.......D.MS./               1     1     1      3

..........A.W.........../ ....5...M../ .M............./ .M.....M.................W....G......../ ...G.    1     1     2      5
..........G......................W......G.../                                                         1     1     2      5

......M.........M.........................W......D.U..........................W.........../           1     1     3      1

..................M.....S./ ....S.....S...:.G........./ ....G..........M..../ .......................   1     1     4      4
.W../                                                                                                 1     1     4      4

..............D.MS..../ ........./ ...........................W./ ........G....../ ..................   1     1     5      5
.........A.......D...........G.......G..........O..M../                                               1     1     5      5

..M.....M.........................G.....MS..........W./ ....M...S........G..W.S...M../                1     1     6      2

.....................MS.........................M...G..../ .................3../ ......M.....M...      1     1     7      3
........./                                                                                            1     1     7      3

..........W........................W...W./                                                            1     1     8      1

................D.............D........A...D....M................W./                                  1     1     9      1

......M...............W.......S..................M.../ ....S....M........../ .............D.M../ .     1     1    10      4
.....M................O........4M---.D./ ..........MS...MS....../                                     1     1    10      5

...MS........G...M........W..S../ ..MS......G...............W..../ ........S....../                    1     1    11      3

........AF.............M.....W....MS..../ ...S.............D.S../                                      1     1    12      2

.....U..........S............D./                                                                      1     1    13      1

.U.....W.........MO..S......S..W./                                                                    1     1    14      1

........A...M.......M../                                                                              1     1    15      1

....M..............MC...........AF./                                                                  1     1    16      1

......M..........M............MO................AF./                                                  1     1    17      1

.....M....MO......S.O..../                                                                            1     1    18      1

..F......S..S..../ ..........S..............5../                                                      1     1    19      2

S........M../                                                                                         1     1    20      1

S.......F....C.W./ .......DG............A........M.F../ .....S......AF.....G..US.D...../               1     1    21      3
```

```
                                                                                                                        PAGE   7
S....C.../ ..............G1..F............../ ........M.......S..........W........G1./ .........1..         1    1    22     4
O..........S..../ .........S....O./                                                                          1    1    22     5

.S.D....E.F...../ ...................../ ..........AF.W...../ ..MS.....3......./                             1    1    23     4

.......S....................M...G.........S...........M../                                                  1    1    24     1

...S..../ .............5P................./'                                                                 1    1    25     2

....P1.W......MS....G..A.........M...S......5..M../ ...........W..D......./ ..S......F.............         1    1    26     3
...D..AF.M../ S.........A.......AF./ .......S...............S.........S./                                   1    1    26     5

.........MS...G..A........W.........C..AF.....M......A......S......A.........A.......MF...W.....S....        1    1    27     1
../                                                                                                         1    1    27     1

.............MS.............../                                                                             1    1    28     1

MS...................D....W.A../                                                                            1    1    29     1

....D......A.........................W...............M..../                                                 1    1    30     1

...S........./                                                                                              1    1    31     1

.....S......../ ...G..W..............,W...........W.....................2./ ............./                   1    1    32     3

....G..A......M......MO...W.............AF./ ............/ .............A......M.......MO..S../              1    1    33     3

.....MS........W./ ..S................W....M.PF......W...M.E................./                              1    1    34     2

.....W..S.G..................../ .....S.........A..W..................P..E...........S......AF...S..         1    1    35     2
......S..............C.....MO./                                                                             1    1    35     2

......AF.MS................P.S1.W...........W.........D./                                                   1    1    36     1

.......MS.........W./ ...SLG.................D./                                                            1    1    37     2

....MS..........W........SD..AF.............O.....S.......AF./                                              1    1    38     1

...............SO...........F./ .........S........AF...........A../                                         1    1    39     2

.........AF.MS......W..../ .....S.................C.M../                                                    1    1    40     2

...S......MS....M.P1......E.M..2../ .............W.....AF...M...E.....C..W...../                             1    1    41     2

......MS./ ...................D.......AF...W./                                                              1    1    42     2

........MS............M.P....M..........M.........SO..AF.W..............O./ .....S........1.......A../     1    1    43     2
....S.............C.W.........A..W.......AF.W../                                                            1    1    43     3

.........MS.../                                                                                             1    1    44     1

.MS............W..A........../ ....M..M.P..E.2......W..........S.........W.............A......S........     1    1    45     2
...W./                                                                                                      .    1    45     2

.................M.P...............C.W....A......./                                                         1    1    46     1

..D...W........W.MS./                                                                                       1    1    47     1

................W./ ......W.....W./ ................C....C.........E......../                               1    1    48     3
```

```
                                                                                                                                          PAGF   8
...........W./ ...........W........................O./                                                                        1    1    49     2

MS....W..S.......O.....AP............E..AF.............W1....../                                                              1    1    50     1

............................W................/                                                                                1    1    51     1

........F............/ ..W..................................../                                                               1    1    52     2

............................W........./ .............................../ .........................                            1    1    53     3
.............../ ....................../                                                                                      1    1    53     4

....W..S...E.F....../ ..........W...............W..............C........F.MO......W./ ....M....DC..                           1    1    54     3
......./                                                                                                                      1    1    54     3

..MS..W.........../ ......G..W...............M../ .S............................F.--.O.............                           1    1    55     3
.......GS../                                                                                                                  1    1    55     3

....S...........MS....S../ ........MO..../ ........MO.........M..../ .............W.......O.../ ...                          1    1    56     5
S.........C..........C./                                                                                                      1    1    56     5

.D.S.....................MO./ ....MC........C..................../  ....MO........................                             1    1    57     3
.S../                                                                                                                         1    1    57     3

S..S....SO......W....../ ........1...W..........................W..../ ....W......1.W.....S.F.....                           1    1    58     3
....../ .F..........SO................................/ .SO.....AF.........F...C...............O..                           1    1    58     5
..F...................../                                                                                                     1    1    58     5

......SC.....MO..F.....W............./                                                                                        1    1    59     1

... ......S...AF./ .....W..S.................AF....O................./ ......................../                              1    1    60     3

......S.......S......W...W.......D./ .......M...AF./                                                                          1    1    61     2

....................................S../ ..........F......O...S......................1...........W.                          1    1    62     2
.......S......AF./                                                                                                            1    1    62     2

...S....W.......U.............O.../ ............................../                                                           1    1    63     2

..S.....W..........S............../ ............O......AF.....O../                                                            1    1    64     2

......MS................../                                                                                                   1    1    65     1

...G..MS.......S.M................../                                                                                         1    1    66     1

.....W...........W...1........P./                                                                                             1    1    67     1

...........F.....W.........../ ........M.....P../                                                                             1    1    68     2

...W....P....MS.P................/                                                                                            1    1    69     1

.................W......./ .....W...P..W............................./                                                        1    1    70     2

..........M.....MS.P./ .......M........P......M..../ ,........n...W.......M......P....1...W........                           1    1    71     3
......../                                                                                                                     1    1    71     3

....F.........M....C./ ..M......../                                                                                           1    1    72     2

.....5..W....P........F............/ .....W..........5./ ......W..P.............../                                           1    1    73     3
```

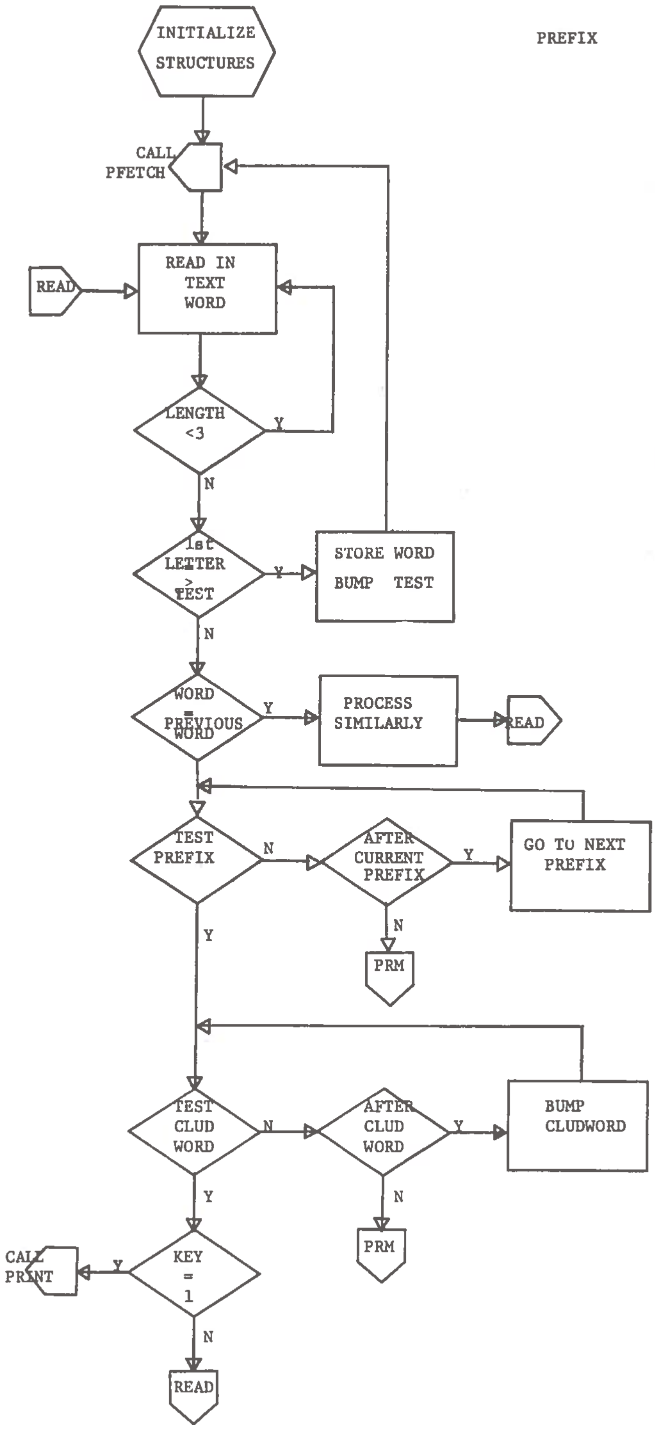
APPENDIX F

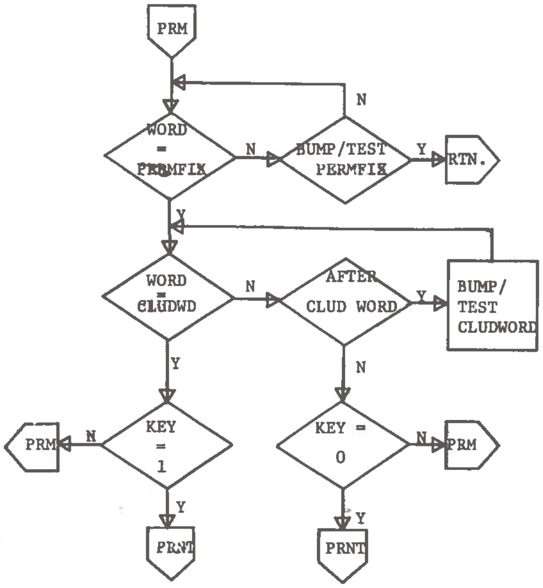
PREFIX Program and Table Listing

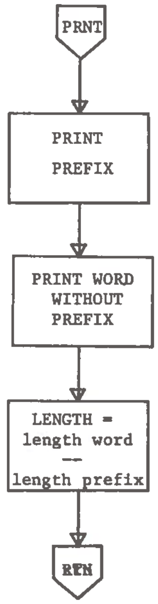
by

John B. Smith

PREFIX


INITIALIZE
STRUCTURES
CALL
PFETCH
READ
READ IN
TEXT
WORD
LENGTH
<3
Y
N
1st
LETTER
>
TEST
Y
STORE WORD
BUMP TEST
N
WORD
=
PREVIOUS
WORD
Y
PROCESS
SIMILARLY
READ
TEST
PREFIX
N
AFTER
CURRENT
PREFIX
Y
GO TO NEXT
PREFIX
N
PRM
Y
TEST
CLUD
WORD
N
AFTER
CLUD
WORD
Y
BUMP
CLUDWORD
N
PRM
Y
KEY
=
1
Y
CALL
PRINT
N
READ


PRM
N
WORD
=
PERMFIX
N
BUMP/TEST
PERMFIX
Y
RTN.
Y
WORD
=
CLUDWD
N
AFTER
CLUD WORD
Y
BUMP/
TEST
CLUDWORD
Y
N
KEY
=
1
N
PRM
Y
PRNT
KEY =
0
N
PRM
Y
PRNT


PRNT
PRINT
PREFIX
PRINT WORD
WITHOUT
PREFIX
LENGTH =
length word
—
length prefix
RTN

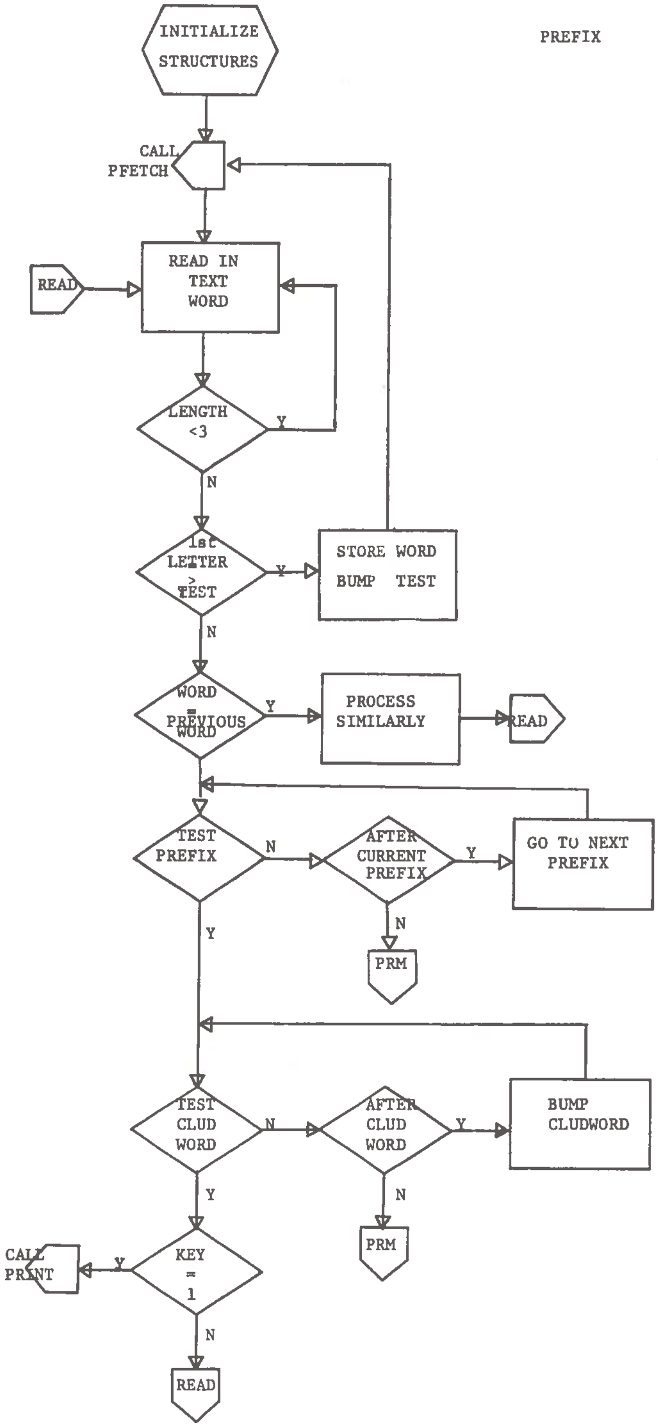
APPENDIX F

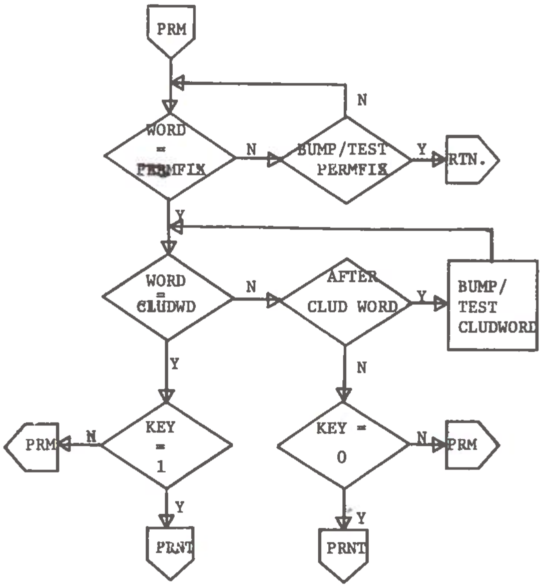
PREFIX Program and Table Listing

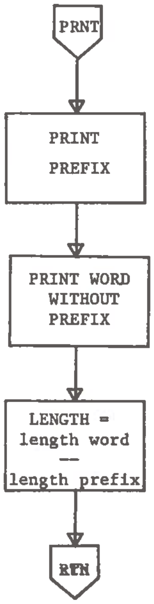
by

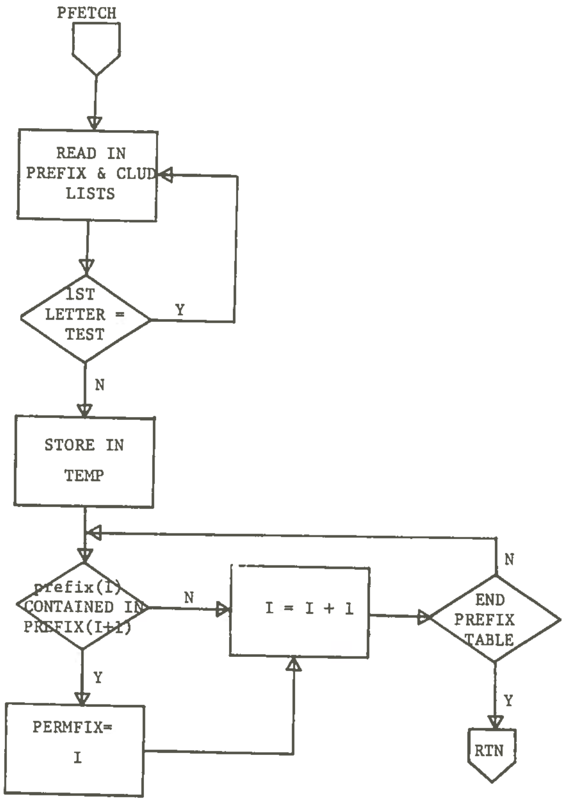
John B. Smith

PREFIX


INITIALIZE STRUCTURES
CALL PFETCH
READ
READ IN TEXT WORD
LENGTH <3
Y
N
1st LETTER > TEST
Y
STORE WORD BUMP TEST
N
WORD = PREVIOUS WORD
Y
PROCESS SIMILARLY
READ
TEST PREFIX
N
AFTER CURRENT PREFIX
Y
GO TO NEXT PREFIX
N
PRM
Y
TEST CLUD WORD
N
AFTER CLUD WORD
Y
BUMP CLUDWORD
N
PRM
Y
KEY = 1
Y
CALL PRINT
N
READ


PRM
N
WORD = PERMFIX
N
BUMP/TEST PERMFIX
Y
RTN.
Y
WORD = CLUDWD
N
AFTER CLUD WORD
Y
BUMP/ TEST CLUDWORD
Y
N
KEY = 1
N
PRM
Y
PRNT
KEY = 0
N
PRM
Y
PRNT


PRNT
PRINT PREFIX
PRINT WORD WITHOUT PREFIX
LENGTH = length word — length prefix
RTN


PFETCH
READ IN
PREFIX & CLUD
LISTS
1ST
LETTER =
TEST
Y
N
STORE IN
TEMP
N
prefix(I)
CONTAINED IN
PREFIX(I+1)
N
I = I + 1
END
PREFIX
TABLE
Y
PERMFIX=
I
Y
RTN

```
STMT LEVEL NEST
  1                 PREFIX:  PROCEDURE OPTIONS(MAIN);

                                      /*   *****************************************     */
                                      /*   PREFIX IS A GENERAL PURPOSE PROGRAM USED      */
                                      /*   TO ANALYZE USAGE OF ENGLISH PREFIXES.         */
                                      /*   IT DOES THIS BY STRIPPING A WORD OF ITS       */
                                      /*   PREFIX AND REPRODUCING THE ROOT  FORM OF      */
                                      /*   THE WORD WITH ITS DETACHED PREFIX.            */
                                      /*   THE SPECIFIC FUNCTION OF PREFIX IN THE        */
                                      /*   VIA PACKAGE IS TO CREATE DUPLICATES OF WORDS  */
                                      /*   WITH PREFIX DETACHED AND TO INSERT  THESE     */
                                      /*   FORMS INTO THE DATA STREAM ALONG WITH THE     */
                                      /*   ORIGINAL FORM OF THE WORD.  THE RESULT IS     */
                                      /*   THAT THE FREQUENCY COUNTS OF THE ROOT FORM    */
                                      /*   OF THE WORD WILL BE MODIFIED, PERHAPS         */
                                      /*   FORCING THE TOTAL OVER PARAMETERS KEYING      */
                                      /*   OTHER ANALYTIC STEPS.                         */
                                      /*                                                 */
                                      /*   ENGLISH PREFIXES, ARRANGED IN ALPHABETIC      */
                                      /*   ORDER ARE LOADED INTO A STRUCTURE ALONG WITH  */
                                      /*   A LIST OF WORDS THAT ARE EITHER EXCLUSION     */
                                      /*   LISTS FOR A PARTICULAR PREFIX OR INCLUSION    */
                                      /*   FORMS, I.E. WORDS THAT DO HAVE                */
                                      /*   LEGITIMATE PREFIXES ATTACHED.  THE NATURE OF  */
                                      /*   THE LIST IS DETERMINED BY A KEY ALSO LOADED   */
                                      /*   INTO THE STRUCTURE.                           */
                                      /*   TEXT WORDS OR WORDS UNDER  ANALYSIS ARE ALSO  */
                                      /*   ARRANGED IN ALPHABETIC ORDER AND ARE EX-      */
                                      /*   AMINED ONE AT A TIME.  IF THE FIRST X CHARAC- */
                                      /*   TERS OF A WORD (CORRESPONDING TO THE LENGTH   */
                                      /*   OF THE PREFIX)  MATCH THE PREFIX, THEN A      */
                                      /*   SEARCH IS MADE OF THE 'CLUD' LIST ASSO-       */
                                      /*   CIATED WITH THE PREFIX.  IF A MATCH IS FOUND  */
                                      /*   THEN THE FIRST X CHARACTERS ARE STRIPPED OR   */
                                      /*   NOT DEPENDING UPON WHETHER THE LIST IS AN     */
                                      /*   INCLUSION OR AN EXCLUSION LIST.               */
                                      /*   *******************************************   */


                                      /*  MAIN STRUCTURE THAT HOLDS PREFIXES, CLUD LIST  */
                                      /*  AND KEY.  THE PROGRAM READS IN ALL PREFIXES    */
                                      /*  FOR A PARTICULAR LETTER OF THE ALPHABET.       */
  2      1          DCL 01 PTABLE (35),
                          02 PRFIX CHAR(8) VARYING,
                          02 KEY FIXED DEC(1),
                          02 CLUDWD(300) CHAR(18) VARYING;


                                      /*  TEMP STORES THE FIRST PREFIX OF THE NEXT       */
                                      /*  LETTER OF THE ALPHABET.  TEMP BECOMES PTABLE   */
                                      /*  (1) WHEN THE STRUCTURE IS NEXT LOADED.         */
  3      1          DCL 01 TEMP,
                          02 TEMPFIX CHAR(3) VARYING INITIAL (' '),
                          02 TEMPKEY FIXED DEC(1),
                          02 TEMPCLD(300) CHAR(18) VARYING;
```

```
        PREFIX:  PROCEDURE OPTIONS(MAIN);

STMT LEVEL NEST

                                  /*  LETR = ARRAY HOLDING ALPHABET FOR TESTS AND     */
                                  /*  CONTROL OF MAIN DO LOOP.                        */
  4     1          DCL LETR  (26)  CHAR(1);

  5     1          DCL TOP FIXED DEC(2)  INITIAL(0);
                                  /*  WORD BEING TESTED FOR PREFIX                    */
  6     1          DCL WKWORD CHAR(18);

  7     1          DCL ALLWORD CHAR(13);
  8     1          DCL LSTWORD CHAR(18);
  9     1          DCL REJECT CHAR(18);

 10     1          DCL 01 DUPREC,
                         02 DUPFIX CHAR(8)  VARYING,
                         02 DUPWRD CHAR(13)  VARYING;

 11     1          DCL CH CHAR(1);
 12     1          DCL N FIXED DEC(2)  INITIAL(1);
 13     1          DCL P FIXED DEC(3);
 14     1          DCL X FIXED DEC(1);
 15     1          DCL PNO FIXED DEC(3)  INITIAL(1);
 16     1          DCL Y FIXED DEC(2);
 17     1          DCL PERMFIX  (10)  FIXED DEC(2);
 18     1          DCL PERMOLD  (10)  CHAR(18)  VARYING;
 19     1          DCL FRSTL FIXED DEC(3)  INITIAL(0);
 20     1          DCL TEMPWD CHAR(18)  INITIAL(' ');
 21     1          DCL COUNTT FIXED DEC(5)  INITIAL(0);
 22     1          DCL COUNTS FIXED DEC(5)  INITIAL(0);
 23     1          DCL LAST CHAR(18)  INITIAL(' ');
 24     1          ON ENDFILE(LIND)   GO TO OUT;


 26     1          PUT PAGE;
 27     1          PUT EDIT('WORD', 'PREFIX', 'STEM + ENDINGS','PREFIX OUTPUT:   PAGE ',
                   PNO) (COL(1), A, COL(20), A, COL(30), A, COL(80), A, F(3));
 28     1          PNO = PNO + 1;
 29     1          PUT SKIP(2);
 30     1          ON ENDPAGE BEGIN;
 32     2          PUT PAGE;
 33     2          PUT EDIT('WORD', 'PREFIX', 'STEM + ENDINGS','PREFIX OUTPUT:  PAGE ',
                   PNO) (COL(1), A, COL(20), A, COL(30), A, COL(80), A, F(3));
 34     2          PNO = PNO + 1;
 35     2          PUT SKIP(2);
 36     2          END;

 37     1          GET EDIT(LETR, CH) (26 A(1), X(53), A(1));

                                /*  MAIN DO LOOP THAT CONTROLS TEST LETTER OF
                                /*  ALPHABET.
 38     1          DO I = 1 TO 26;
 39     1     1    IF LETR(I) = 'J' THEN GO TO BUMP;

 41     1     1    N = 1;
 42     1     1    P = 1;
```

```
      PREFIX:  PROCEDURE OPTIONS(MAIN);                                                        PAGE     4

STMT LEVEL NEST

                                       /*  CALLS SUBPROCEDURE THAT LOADS PREFIX TABLES.    */
  43     1    1    CALL PFETCH;

                                       /*  TESTS TO SEE IF CURRENT BATCH OF PREFIXES       */
                                       /*  CORRESPOND WITH CURRENT TEST LETTER.            */


                                       /*  TEMPWORD IS LAST WORD READ BEFORE PREFIX        */
                                       /*  STRUCTURE LOADED.  ID DID NOT MATCH LETTER(I)=  */
                                       /*  AT THAT TIME.                                   */
  44     1    1    A:  IF TEMPWD ¬= ' ' THEN DO;
  45     1    1    WKWORD = TEMPWD;
  47     1    1    TEMPWD = ' ';
  48     1    1    END;

  49     1    1    ELSE DO;
  50     1    1    WKWORD = ' ';
  51     1    1    DCL JUNK1 FIXED DEC(2) INITIAL(0);
  52     1    1    DCL JUNK2 CHAR(16);
  53     1    1    GET FILE(LIND) EDIT(JUNK1, JUNK2, ALLWORD) (F(2), A(16), A(18));
  54     1    1    COUNTT = COUNTT + 1;
  55     1    1    PUT FILE(ADD)  EDIT(JUNK1, JUNK2, ALLWORD) (F(2), A(16), A(18));
  56     1    1    DO I3 = 18 TO 1 BY -1;
  57     1    2    IF SUBSTR(ALLWORD,I3,1) ¬= ' ' THEN GO TO OUTA;
  59     1    2    END;


                                       /*  PROGRAM DISCARDS ALL WORDS WITH FEWER THAN 4    */
                                       /*  LETTERS IN THEM, ASSUMING SUCH WORDS DO NOT     */
                                       /*  HAVE LEGITIMATE PREFIXES.                       */
  60     1    1    OUTA:
  61     1    1    IF I3 <= 3 THEN GO TO A;
  62     1    1    WKWORD = SUBSTR(ALLWORD,1,I3);
  63     1    1    END;

                                       /*  CHECKS CURRENT WORD FOR LETTER MATCH PREFIXES   */
  64     1    1    IF SUBSTR(WKWORD,1,1) > LETR(I)
  65     1    1    THEN DO;
  66     1    1    TEMPWD = WKWORD;
  67     1    1    GO TO BUMP;
  68     1    1    END;

                                       /*  TESTS TO SEE IF CURRENT WORD IDENTICAL TO       */
                                       /*  LAST WORD.  IF SO AND IF LAST WORD DID NOT      */
                                       /*  HAVE AN ALLOWABLE PREFIX, THEN THIS WORD IS     */
                                       /*  SKIPPED ALSO.                                   */
  69     1    1    IF WKWORD = REJECT THEN GO TO A;

                                       /*  IF LAST WORD STRIPPED OF PREFIX, THEN           */
                                       /*  CURRENT WORD IS STRIPPED WITHOUT GOING          */
                                       /*  THROUGH ENTIRE PROCEDURE.                       */
```

```
        PREFIX:  PROCEDURE OPTIONS(MAIN);                                                          PAGE     5

STMT LEVEL NEST

 71      1    1    IF WKWORD = LSTWORD THEN DO;
 73      1    1    CALL PRINT;
 74      1    1    GO TO A;
 75      1    1    END;



                                        /*  SINCE WORDS AND PREFIXES IN ALPHABETICAL          */
                                        /*  ORDER, LAST PREFIX ADDRESS IS STORED SO           */
                                        /*  THAT SEARCH CONTINUES FROM HERE.                  */
 76      1    1    FRSTL = N;

 77      1    1    IF WKWORD = REJECT THEN GO TO SKIP1;
 79      1    1    IF WKWORD = LSTWORD THEN GO TO SKIP1;
 81      1    1    PUT EDIT(WKWORD) (SKIP, COL(1), A);
 82      1    1    SKIP1:
                   X = 1;

 83      1    1    DO L = N TO TOP;

                                        /*  IF FIRST X LETTERS PAST CURRENT PREFIX,           */
                                        /*  SKIPS TO NEXT PREFIX.                             */
 84      1    2    IF SUBSTR(WKWORD,1,LENGTH(PRFIX(L))) > PRFIX(L) THEN GO TO BUMPL;

                                        /*  WHEN MATCH OF FIRST X LETTERS WITH PREFIX IS      */
                                        /*  FOUND SEARCH IS MADE OF CLUD LIST.                */
 86      1    2    IF SUBSTR(WKWORD,1,LENGTH(PRFIX(L)))  = PRFIX(L)
 87      1    2    THEN DO M = P TO 300 WHILE (PTABLE(L).CLUDWD(M)   = ' ');

                                        /*  TESTS TO SEE THAT WORD IS NOT PAST CLUD WORD.  */
 88      1    3    IF PTABLE(L).CLUDWD(M) > SUBSTR(WKWORD,1,LENGTH(PTABLE(L).CLUDWD(M)))
 89      1    3    THEN DO;
 90      1    3    P = M;
 91      1    3    IF SUBSTR(WKWORD,1,LENGTH(PRFIX(L + 1))) >= PRFIX(L + 1) THEN GO TO
                   BUMPL;
 93      1    3    ELSE GO TO PRMLOOP;
 94      1    3    END;

                                        /*  LOCATION OF LAST CLUD MATCH.  NEXT SEARCH         */
                                        /*  BEGINS HERE.                                      */
 95      1    3    IF SUBSTR(WKWORD,1,LENGTH(PTABLE(L).CLUDWD(M))) =
                   PTABLE(L).CLUDWD(M)
 96      1    3    THEN DO;
 97      1    3    P = M;

                                        /*  WHEN MATCH WITH CLUD WORD IS FOUND, PROGRAM       */
                                        /*  CHECKS KEY TO DETERMINE WHETHER THE LIST IS       */
                                        /*  INCLUSION OR EXCLUSION LIST.                      */
 98      1    3    IF KEY(L) = '0' THEN DO;
100      1    3    REJECT = WKWORD;
101      1    3    GO TO A;
102      1    3    END;
```

```
STMT LEVEL NEST

 103     1    3    DO;
 104     1    3    Y = L;
                                      /*  PRINT IS SUBPROCEDURE THAT PRINTS WORD WITH    */
                                      /*  PREFIX REMOVED.                                */
 105     1    3    CALL PRINT;
 106     1    3    N=L;
 107     1    3    GO TO  A;
 108     1    3    END;

 109     1    3    END;

 110     1    3    END;


 111     1    2    DO;
 112     1    2    IF SUBSTR(WKWORD,1,LENGTH(PRFIX(L + 1))) >= PRFIX(L + 1) THEN GO TO
                   BUMPL;
 114     1    2    ELSE GO TO PRMLOOP;
 115     1    2    END;
 116     1    2    BUMPL:  P = 1;
 117     1    2    END;


                                      /*  ******************************************* */

                                      /*  SINCE SOME PREFIXES OVERLAP WITH PREFIXES      */
                                      /*  THAT FOLLOW, IT IS POSSIBLE FOR WORDS THAT     */
                                      /*  APPEAR AFTER THE SECOND PREFIX MAY ACTUALLY    */
                                      /*  HAVE THE FIRST PREFIX.  HOPEFULLY AN EXAMPLE   */
                                      /*  WILL HELP.                                     */
                                      /*  ATYPICAL HAS A LEGITIMATE A-PREFIX;  HOWEVER   */
                                      /*  IT WOULD COME AFTER ALL WORDS WITH AD/PREFIX-  */
                                      /*  ES.  IN ORDER NOT TO LOSE THESE WORDS, A LIST  */
                                      /*  OF ALL SUCH PREFIXES THAT OVERLAP THE          */
                                      /*  FOLLOWING PREFIX IS MADE.  WHEN NO MATCH IS    */
                                      /*  FOUND WITH THE CURRENT PREFIX IN THE NORMAL    */
                                      /*  PROCEDURE, THE PROGRAM JUMPS DOWN TO THIS      */
                                      /*  LOOP AND TESTS WORDS AGAINST THESE             */
                                      /*  OVERLAPPING PREFIXES.                          */
 118     1    1    PRMLOOP:  DO;
                                      /* CHECKS EACH PREFIX AGAINST WORD                 */
 119     1    1    DO X = 1 TO 10 WHILE(PERMFIX(X) ¬= 0);
 120     1    2    IF SUBSTR(WKWORD,1,LENGTH(PRFIX(PERMFIX(X))))¬= PRFIX(PERMFIX(X)) |
 121     1    2    SUBSTR(WKWORD,1,LENGTH(PERMCLD(X))) > PERMCLD(X) THEN GO TO ENDX;

                                      /* IF MATCH IS  FOUND, CHECKS CLUD LIST            */
 122     1    2    DO J = 1 TO 300 WHILE(PTABLE(PERMFIX(X)).CLUDWD(J) <= PERMCLD(X));
 123     1    3    IF SUBSTR(WKWORD,1,LENGTH(PTABLE(PERMFIX(X)).CLUDWD(J))) <
 124     1    3    PTABLE(PERMFIX(X)).CLUDWD(J) THEN GO TO ENDX;
 125     1    3    IF SUBSTR(WKWORD,1,LENGTH(PTABLE(PERMFIX(X)).CLUDWD(J))) =
 126     1    3    PTABLE(PERMFIX(X)).CLUDWD(J) THEN DO;
```

```
        PREFIX:  PROCEDURE OPTIONS(MAIN);

STMT LEVEL NEST

 127      1    3     IF KEY(PERMFIX(X)) = '0' THEN DO;      /* CHECKS IN/EX-CLUSION LIST   */
 129      1    3     REJECT = WKWORD;
 130      1    3     GO TO A;
 131      1    3     END;

 132      1    3     ELSE DO;
 133      1    3     Y = PERMFIX(X);
 134      1    3     IF N < PERMFIX(X) THEN N = PERMFIX(X);  /*START SUCCESSIVE SCANS HERE*/
 136      1    3     CALL PRINT;
 137      1    3     GO TO A;
 138      1    3     END;

 139      1    3     END;

 140      1    3     ENDJ:  IF PTABLE(PERMFIX(X)).CLUDWD(J) = PERMCLD(X)
 141      1    3     THEN DO;
 142      1    3     GO TO ENDX;
 143      1    3     END;

 144      1    3     END;

 145      1    2     ENDX:  END;


 146      1    1     IF L > TOP THEN L = TOP;

 148      1    1     DO L2 = FRSTL TO L;

 149      1    2     B:  IF KEY(L2) = '0'
 150      1    2     THEN IF SUBSTR(WKWORD,1,LENGTH(PRFIX(L2))) = PRFIX(L2)
 151      1    2     THEN DO;
 152      1    2     Y = L2;
 153      1    2     IF N < L2 THEN N = L2;
 155      1    2     CALL PRINT;
 156      1    2     GO TO A;
 157      1    2     END;

 158      1    2     END;


 159      1    1     DO K = 1 TO 10 WHILE(PERMFIX(X) ¬= 0);

 160      1    2     IF KEY(PERMFIX(X)) = 0
 161      1    2     THEN IF SUBSTR(WKWORD,1,LENGTH(PRFIX(PERMFIX(X)))) = PRFIX(PERMFIX(X))
 162      1    2     THEN DO;
 163      1    2     Y = PERMFIX(X);
 164      1    2     IF N < PERMFIX(X) THEN N = PERMFIX(X);
 166      1    2     CALL PRINT;
 167      1    2     GO TO A;
 168      1    2     END;

 169      1    2     END;
```

```
STMT LEVEL NEST

 170     1    1     REJECT = WKWORD;
 171     1    1     GO TO A;
 172     1    1     END;


 173     1    1     BUMP:  END;



                                   /* PRINT IS THE SUBPROCEDURE THAT CREATES A DUP-     */
                                   /* LICATE RECORD AND PRINTS THE RECORD.              */
 174     1          PRINT: PROCEDURE;
 175     2          IF SUBSTR(WKWORD,LENGTH(PREFIX(Y)) + 1,1) = '-' THEN DO;
 177     2          DUPFIX = SUBSTR(WKWORD,1,LENGTH(PREFIX(Y)) +1);
 178     2          DUPWRD = SUBSTR(WKWORD,LENGTH(PREFIX(Y)) + 2);
 179     2          GO TO PRNT;
 180     2          END;
 181     2          DUPFIX = SUBSTR(WKWORD,1,LENGTH (PREFIX(Y)));
 182     2          DUPWRD = SUBSTR(WKWORD,((LENGTH(PREFIX(Y))) + 1));
 183     2          PRNT:
                                   /* LENGTH OF THE WORD IS RECOMPUTED FOR SUFFIX      */
                    JUNK1 = JUNK1 - LENGTH(PREFIX(Y));
 184     2          PUT FILE(ADD) EDIT(JUNK1, JUNK2, DUPWRD) (F(2), A(16), A(18));
 185     2          IF WKWORD = LSTWORD THEN GO TO SKIP2;
 187     2          PUT EDIT(DUPFIX, DUPWRD)(SKIP(0), COL(20), A, COL(30), A);
 188     2          LSTWORD = WKWORD; /* SUCCEEDING WORDS TESTED TO AVOID RECOMPUTING    */
 189     2          SKIP2:
                    COUNTS = COUNTS + 1;         /* COUNT OF WORDS WITH PREFIXES KEPT     */
 190     2          END PRINT;



                                   /* PFETCH IS THE SUBPROCEDURE THAT BUILDS THE       */
                                   /* PREFIX TABLES.  IT ALSO DETERMINES WHEN A        */
                                   /*  PREFIX IS 'CONTAINED' IN THE SUCCEEDING PREFIX */
                                   /* SO THAT THESE PREFIXES MAY BE USED IN PERMFIX.  */
 191     1          PFETCH: PROCEDURE;
 192     2          ON ENDFILE(PFIX) GO TO PTESTP;
 194     2          DCL PREFIX1 CHAR(3);
 195     2          DCL TCLUD CHAR(18) INITIAL(' ');

                                   /* CLEARS PREFIX AND CLUD LISTS.                    */
 196     2          DO J = 1 TO 35;
 197     2    1     PREFIX(J) = ' ';
 198     2    1     CLUDWD(J,*) = ' ';
 199     2    1     END;
 200     2          TOP = 0;
```

```
STMT LEVEL NEST

 201     2           PERMPFIX = 0;
 202     2           PERMCLD = ' ';

 203     2           DO J = 1 TO 35;

                                               /* PROGRAM SAVES THE LAST PREFIX WITH CLUD LISTS    */
                                               /* THAT WAS READ IN BUT FOUND TO COME IN LATER      */
                                               /* ALPHABETICAL SEQUENCE THAN THE CURRENT           */
                                               /* PROCESSING LETTER.  THIS BECOMES PREFIX(1)       */
                                               /* OF THE CURRENT PREFIXES.                          */
 204     2    1      IF TEMP.TEMPFIX ¬= ' ' THEN DO;
 206     2    1      PTABLE(1) = TEMP;
 207     2    1      TEMP.TEMPFIX = ' ';
 208     2    1      TEMP.TEMPCLD = ' ';
 209     2    1      GO TO TEST;
 210     2    1      END;

 211     2    1      GET FILE(PFIX) EDIT(PREFIX1, KEY(J), CH) (A(8), X(3), F(1), X(67),
                     A(1));

                                               /* PREFIX IS STORED IN VARYING CHAR. SLOT SO THAT  */
                                               /* THE LENGTH OF THE PREFIX WILL BE AVAILABLE FOR  */
                                               /* SUBSTRING PARAMETER WHEN CHECKING WKWORD FOR    */
                                               /* FOR MATCH.                                       */
 212     2    1      DO II = 8 TO 1 BY -1;
 213     2    2      IF SUBSTR(PREFIX1,II,1) ¬= ' ' THEN GO TO OUTT;
 215     2    2      END;
 216     2    1      OUTT: PRFIX(J) = SUBSTR(PREFIX1,1,II);

 217     2    1      DO K = 1 TO 300;
 218     2    2      GET FILE(PFIX) EDIT(TCLUD, CH)(X(2), A(18), X(59), A(1));

 219     2    2      DO III = 18 TO 1 BY -1;
 220     2    3      IF SUBSTR(TCLUD,III,1) ¬= ' ' THEN GO TO OUTCLD;
 222     2    3      END;
 223     2    2      OUTCLD:  PTABLE(J).CLUDWD(K) = SUBSTR(TCLUD,1,III);

 224     2    2      IF PT[illegible]).CLUDWD(K) = 'END CLUD'
 225     2    2      THEN DO,
 226     2    2      PTABLE(J).CLUDWD(K) = ' ';
 227     2    2      GO TO TEST;
 228     2    2      END;

 229     2    2      IF SUBSTR(PTABLE(J).CLUDWD(K),1,1) = ''''
 230     2    2      THEN DO;
 231     2    2      PTABLE(J).CLUDWD(K) = SUBSTR(PTABLE(J).CLUDWD(K),2,(III-2));
 232     2    2      GO TO ENDK;
 233     2    2      END;

 234     2    2      ENDK:  END;

                                               /* STORES PREFIX AND CLUD LIST FOR NEXT CONST. OF  */
                                               /* PTABLES.                                          */
```

```
        PREFIX:  PROCEDURE OPTIONS(MAIN);                                                                      PAGE    10

STMT LEVEL NEST

 235     2    1    TEST:  IF SUBSTR(PRFIX(J),1,1) ¬= LETR(I)
 236     2    1    THEN DO;
 237     2    1    TEMP = PTABLE(J);
 238     2    1    GO TO PTESTF;
 239     2    1    END;

 240     2    1    TOP = TOP + 1;
 241     2    1    END;


 242     2         PTESTF: X = 1;
                                       /* TESTS TO SEE IF PREFIX IS 'CONTAINED IN' SUC-    */
                                       /* CEEDING PREFIX.                                  */
 243     2         DO J = 1 TO TOP;

 244     2    1    IF SUBSTR(PRFIX(J),1,LENGTH(PRFIX(J))) = SUBSTR(PRFIX(J + 1),1,
 245     2    1    LENGTH(PRFIX(J))) THEN DO;
 246     2    1    PERMFIX(X) = J;

 247     2    1    DO L1 = 1 TO 300;
 248     2    2    IF PTABLE(J).CLUDWD(L1) ¬= ' ' THEN GO TO L1END;

 250     2    2    ELSE DO;
 251     2    2    PERMCLD(X) = PTABLE(J).CLUDWD(L1 - 1);
 252     2    2    X = X + 1;
 253     2    2    GO TO JEND;
 254     2    2    END;

 255     2    2    L1END:  END;

 256     2    1    END;

 257     2    1    JEND:  END;


 258     2         ENDPTCH:  END PFETCH;



                                       /* COMPUTES THE TOTAL NUMBER OF WORDS WITH PREFIXES*/
                                       /* AND THEIR PROPORTION IN THE TEXT.               */
 259     1         OUT:  PUT EDIT('TOTAL WORDS ', COUNTT)(SKIP(2), A, COL(30), F(6));
 260     1         PUT EDIT('TOTAL WORDS WITH PREFIXES', COUNTS)(SKIP, A, COL(30), F(5));
 261     1         PUT EDIT('PERCENT WITH PREFIXES', 100*COUNTS/COUNTT)
                   (SKIP, A, COL(30), F(5,2));



 262     1         END PREFIX;
```

```
                 1
                          ABED                 (NOT)           ABLOOM          ABLOW           ABLUSH
                          ABOIL                ABLAZE          ACHROM          ACRITICAL       ACYCLIC
                          ADANCE               ACENTRIC        ADREAM          ADRIFT          AFAR
                          AFIELD               ADANGLE         AFLAME          AFLOAT          AFLOW
                          AFLUTTER             AFIRE           AFOUL           AGAPE           AGAZE
                          AGLARE               AFOOT           AGLIMMER        AGLINT          AGLISTEN
                          AGLITTER             AGLEAM          AGROUND         AHISTORIC       AHOLD
                          AHORSE               AGLOW           AHUNT           AKIN            ALIGHT
                          ALIKE                AHUM            ALONE           ALOUD           AMASS
                          AMID                 ALIT            APERIODIC       APHONIC         APLENTY
                          ARIPPLE              AMORAL          AROUSE          ASEETHE         ASEXUAL
                          ASHIMMER             ARISE           ASHIVER         ASHORE          ASIDE
                          ASKEW                ASHINE          ASLEEP          ASLOPE          ASOCIAL
                          ASPHERICAL           ASLANT          ASPREAD         ASQUINT         ASTARE
                          ASTATIC              ASPRAWL         ASTRADDLE       ASTRAY          ASWARM
                          ASWAY                ASTIR           ASYLLABIC       ASYMMETR        ASYNCHRON
                          ASYNTACTIC           ASWIRL          ATHIRST         ATHRILL         ATILT
                          ATINGLE              ATHEIS          ATOP            ATREMBLE        ATWIST
                          ATWITTER             ATIPTOE         AVOUCH          AVOW            AWAKE
                          AWASH                ATYPI           AWHEEL          AWHIRL          AWING
                          AWOKE                AWEARY          END CLUD
                                               AWORK
AB               1                  AWAY FROM
                          ABAXIAL              ABERRANT        ABNEGATE        ABNORM          END CLUD
AC               1                  VAR.OF 'AD'
                          ACCEDE               ACCLAIM         ACCLIMAT        ACCOMPANY       ACCOMPT
                          ACCOUNT              ACCOUPL         ACCREDIT        ACCRESCENT      ACCUMULAT
                          ACCURS               ACCUSTOM        ACQUIESC        END CLUD
AD               1                  AWAY FROM
                          ADJOIN               ADMIX           END CLUD
AERO                                           AIR
                          AEROBATIC            AERODROME       AEROGRAM        AEROLIT         AEROMANC
                          AERONAUT             AEROSOL         END CLUD
AFORE            0                  BEFORE
                          AFOREHAND            END CLUD
AFTER            0                  AFTER
                          'AFTER '             AFTERCLAP       AFTERGAME       AFTERMATH       AFTERWARD
                          END CLUD
AG               1                  TOWARD (TENDENCY,DIRECTION,ADDITION)
                          AGGLOMERATE          AGGRIEVE        END CLUD
AL               1                             VAR OF AD,TOWARD TENDENCY,DIRECTION,ADDIT
                          ALLURE               END CLUD
ALL-             0                  ALL
                          END CLUD
ALLO             1                             OTHER
                          ALLOGRAPH            END CLUD
ALTI             0                             HIGH
```

```
PREFIX      KEY     CLUDLIST

                    ALTIMETRY           ALTITUDE            END CLUD

AMBI        1                 BOTH
                    AMBIDEXT            AMBILATERAL         END CLUD

AMPHI       1                           TWO,BOTH,ON BOTH SIDES
                    AMPHITHEATER        END CLUD

AN          1                 NOT,WITHOUT,LACKING,VAR.OF 'AD',VAR.OF 'ANA'_UP,_
                    ANALPHABETIC        END CLUD

ANDRO       1
                    ANDROCENTRIC        ANDROPHOBIA         END CLUD

ANEMO       1                           WIND
                    ANEMOGRA            ANEMOMET            END CLUD

ANGLO       0                           ENGLISH
                    ANGLOPHIL           ANGLOPHOB           END CLUD

ANT         1                           VAR OF 'ANTI' AGAINST
                    ANTACID             ANTARCTIC           END CLUD

ANTE        1                 BEFORE
                    ANTE-CHRISTIAN      ANTE-DAWN           ANTE-MARRIAGE       ANTE-SPRING         ANTE-TASTE
                    ANTE-WAR            ANTECHAMBER         ANTEDAT             ANTEDILUVIA         ANTEHISTORIC
                    ANTEMERIDIAN        ANTEMUNDANE         ANTENATAL           ANTENUMBER          ANTENUPTIAL
                    ANTEPRANDIAL        ANTEPROHIBITION     ANTEROOM            ANTETYPE            END CLUD

ANTHROPO    1                 HUMAN
                    ANTHROPOCENTRI      ANTHROPOGEN         ANTHROPOGEOGRAPH    END CLUD

ANTI        0                 AGAINST,OPPOSITE OF
                    ANTIBODY            'ANTIC '            ANTICIPA            ANTIDOTE            ANTINOMY
                    ANTIPATH            ANTIPOD             ANTIQU              ANTITYP             END CLUD

AP          1                 VAR.OF AD,VAR.OF APO _ AWAY,DIFFERENT,FROM
                    'APPEND '           APPERTAIN           APPLY               APPOS               APPRESS
                    END CLUD

AR          1                           VAR OF AD BEFORE 'R'
                    ARREAR              END CLUD

ARCH        1                           (CHIEF)
                    ARCH-ENEMY          ARCH-FOE            ARCH-HERE           ARCH-LIAR           ARCH-OPPONENT
                    ARCH-POET           ARCH-TRAITOR        ARCH-VERSIFIER      ARCH-VILLIAN        ARCHANGEL
                    ARCHBISHOP          ARCHCHANCEL         ARCHDEACON          ARCHDIOCESE         ARCHDU
                    ARCHENEMY           ARCHFIEND           ARCHHERE            ARCHPRESBYTER       ARCHPRIEST
                    ARCHSEE             ARCHVILLIAN         END CLUD

ARCHE       1                           (PRIMITIVE)
                    ARCHETYP            END CLUD

ARCHI       1                           (CHIEF)
                    ARCHIDIACONAL       ARCHIEPISCOPA       END CLUD

AT          1                           VAR OF AD
                    ATTEMPER            ATTRACT             ATTRIBUT            ATTUN               END CLUD
```

```
PREFIX      KEY     CLUDLIST

ATMO        1                          AIR
                    ATMOSPHER          END CLUD

AUDIO       0                AUDITORY
                    AUDIOGEN
                    AUDIOLOG (Y)
                    END CLUD                                                  AUDION             AUDIOPHIL

AUTO        0                          (SELF,SAME)
                    AUTOLOG            AUTOMAT            AUTOMETRY           AUTONOM            AUTONYM
                    AUTOCLAVE          AUTOCHTHON         AUTOGRAPH           AUTOPSY            END CLUD

BACK        0                BACK
                    'BACK '            BACKE              BACKING             BACKLOG            BACKSLIDE
                    BACKSTAB           BACKSWEPT          BACKWARD            END CLUD

BE          1                COVER,TO MAKE_,TO DUB,PROVIDED WITH,NO MEANING
                    BE-NIGHTMARED      BEBOOTED           BECHARM             BECLOUD            BECRAWL
                    BECRIPPL           BEDABBL            BEDASH              BEDAUB             BEDAVID
                    BEDAZZL            BEDEVIL            BEDEW               BEDIAMONDED        BEDIM
                    BEDRABBL           BEDRAGGL           BEDRIVEL            BEFLAG             BEFLEA
                    BEFLOWER           BEFOG              BEFOOL              BEFOUL             BEFRIEND
                    BEGEM              BEGLAMOUR          BEGLAR              BEGRIM             BEHEAD
                    BEJESUIT           BEJEWEL            BEKNAVE             BELATE             BELITTL
                    BEMEAN             BEMEDALLED         BEMIRE              BEMOAN             BEMOCK
                    BENIGHT            BEPAINT            BERASCAL            BERHYM             BERIBANDED
                    BERIBBONED         BEPIM              BEROGUE             BESCRIBBL          BESEIG
                    BESLOBBER          BESPANGL           BESPECTACLED        BESPREAD           BESTRADDL
                    BESTRID            BESTROD            BESTUD              BETHINK            BETHOUGHT
                    BEWAIL             BEWEEP             BEWHISKERED         BEWIGGED           BEWITCH
                    BEWRITE            END CLUD

BI          0                TWO,TWICE,VAR.OF 'BIO'
                    BIAS               'BIB '             BIBLE               BIBLIO             BIBULOUS
                    BICIPITAL          'BID '             BIDDABL             BIDDEN             BIDDING
                    BIDDY              BIDE               BIENNIAL            BIER               BIFARIOUS
                    BIFID              'BIG '             BIGAM               BIGOT              BIJOU
                    BILE               BILIOUS            BILK                BILL               BILLET
                    BILLION            BILLOW             BIMESTER            'BIN '             BINAL
                    BINARY             BINAURAL           BIND                BINOCLE            BINOCULAR
                    BIO                BIPOD              BIRCH               BIRD               BIRR
                    BIRTH              'BIS '             BISCUIT             BISECT             BISHOP
                    'BIT '             BITCH              BITE                BITING             BITTEN
                    BITTER             BITUM              BIZARRE             END CLUD

BIBLIO      1                BOOK,BIBLE
                    BIBLIOFILM         BIBLIOMANI         END CLUD

BIN         1                TWO,TWO AT A TIME
                    BINAURAL           BINOCULAR          END CLUD

BOOK        0                          BOOKIE,BOOKING,BOOKISH
                    'BOOK '            BOOKI              BOOKLET             'BOOKS '           END CLUD

BY          0                          (ACCESSORY,PAST,SUBORDINATE,BY THE SIDE)
                    BY-BLOW            BY-WORD            BYE                 BYRNE              BYTE
                    BYWORD             END CLUD
```

```
PREFIX    KEY     CLUDLIST

BYE       1                VAR.OF 'BY'
                  BYELAW          END CLUD

CENTRI    0                       (CENTER)
                  CENTRIC         CENTRIFUGAL     CENTRIOLE       END CLUD

CHRONO    1                       TIME
                  CHRONOGRAPH     CHRONOMET       CHRONOSCOP      END CLUD

CIS       1                       (NEAR SIDE OF)
                  CISATLANTIC     CISLUNAR        END CLUD

CO        1                       VAR OF COM, IN ASSOC WITH
                  CO_ORDIN        CO_SIGN         CO_STAR         CO_WORKER       COACT
                  COADJUT         COADVENTUR      COAXIAL         CODEFEND        COEDIT
                  COEDUCAT        COEQUAL         COETERN         COEX            COFUNCTION
                  COHEIR          COINCIDEN       COINSUR         COMATE          COOPERAT
                  COORDIN         COPARTNER       COTEMPOR        COTENANT        COTERMINOUS
                  END CLUD

COL       1                       VAR OF COM,WITH
                  COLLABORA       COLLAPS         COLLATERAL      COLLEAGUE       COLLOCAT
                  END CLUD

COM       1                       WITH,TOGETHER,IN ASSOC
                  COMMEASUR       COMMEMOR        COMMINGLE       COMMUTA         COMMUTUAL
                  COMPACT         COMMPATERN      COMPATRIOT      COMPEER         COMPOSSIBLE
                  COMPROMIS       END CLUD

CON       1                       VAR OF COM
                  CONCAV          CONCENTRIC      CONCORPORAT     CONCOURSE       CONDENS
                  CONDESCEN       CONFEDERA       CONFIGUR        CONFORM         CONFRONT
                  CONGENIAL       CONGENITAL      CONJOIN         CONJUNCTION     CONJUNCTU
                  CONSEQUEN       CONSOLIDAT      CONSTRAIN       CONSTRICT       CONTEMPOR
                  CONTORTION      'CONTRACT '     END CLUD

COUNTER   0                       (OPPOSITE)
                  COUNTERCHANGE   COUNTERFEIT     COUNTERMAN      COUNTERPART     COUNTERWORD
                  END CLUD

DE        1                       SEPARAT,PRIVATION,REMOV,DESCENT,REVERSAL
                  DE_EMPHASI      DAERAT          DEBASE          DEBRIEF         DEBUG
                  DECAMP          'DECANT '       DECAPITAT       DECENTER        DECENTR
                  DECERTIF        DECLASS         DECOD           DECOLONI        DECOM
                  DECON           DECRESC         DECRY           DECURV          DEDUCT
                  DEFAC           DEFANG          DEFEATUR        DEFEND          DEFLOWER
                  DEFOLIAT        DEFORM          DEFRAUD         DEFROST         DEFUS
                  DEGENERA        DEGLACIA        DEGLAMORIZ      DEGRAD          DEGUM
                  DEHORN          DEHUM           DEHYDRAT        DEICE           DELAMINA
                  DELEGALIZ       DELEGATION      DELIMIT         DELINEAT        DELIST
                  DELOCAL         DELOUS          DEMAGNETI       DEMARK          DEMATERIALI
                  DEMEAN          DEMERIT         DEMILITAR       DEMOBIL         DEMODULAT
                  DEMORALI        DEMOUNT         DENA            DENOMINAT       DENOT
                  DENUMERA        DEODOR          'DEPART '       'DEPARTED '     'DEPARTING
                  DEPEOPLE        DEPERSON        DEPICTUR        DEPLAN          DEPLOY
                  DEPOL           DEPOP           DEPORT          END CLUD

DECA      1                       TEN
```

```
PREFIX      KEY     CLUDLIST

                    DECAGRAM           DECALITER          DECAMETER          END CLUD
DECI        1                          TENTH
                    DECIGRAM           DECILITER          DECIMETER          END CLUD
DEMI        1                          (HALF)
                    DEMIBLOND          DEMIGOD            END CLUD
DI          1                          TWO,DOUBLE
                    DIATOMIC           DICHROM            DISYLLAB           DITHEIS            END CLUD
DIS         0                          APART,AWAY,UTTERLY,PNR
                    DISABUS            DISAFFECT          DISAST             DISBURS            'DISC '
                    DISCER             DISCI              DISCLOSE           DISCORD            DISCREET
                    DISCREPAN          DISCRET            DISCRIM            'DISCUS '          DISCUSS
                    DISDAIN            DISEAS             DISGRUNT           'DISH '           DISHCOVER
                    'DISHES '          DISHEVEL           DISHFUL            DISHRAG            DISHTOWEL
                    DISHWA             DISK               DISMAL             DISMAY             DISMISS
                    DISPARA            DISPATCH           DISPEL             DISPEN             DISPER
                    DISPIRIT           DISPLAY            DISPOSA            DISPOSE            DISPOSI
                    DISPOSUR           DISPREAL           DISPUT             DISRUPT            'DISSECT '
                    DISSECTED          DISSEMINAT         DISSEN             DISSERT            DISSIIDEN
                    DISSIPAT           DISSOLUT           'DISSOLVE '        'DISSOLVI '        DISSONAN
                    DISTAFF            DISTAIN            DISTAL             DISTAN             DISTEN
                    DISTI              DISTORT            DISTRACT           DISTRAUGHT         DISTRESS
                    DISTRICT           DISTURB            DISUAD             DISUASIVE          DISYLLAB
                    END CLUD
DOWN        0                          (DOWN)
                    'DOWN '            DOWN-TO-EARTH      DOWNPAYMENT        DOWNRIGHT          DOWNSTAGE
                    DOWNTIME           DOWNTOWN           DOWNWARD           DOWNY              END CLUD
E           1                          VAR OF 'EX' UTTERLY,ETC.
                    'EDUCE '           EDUCT              ELABOR             ELAPS              ELOCUTION
                    ELOPE              ELUCIDAT           EMASCULAT          'EMERGE '          EMERGED
                    'EMERGENT '        'EMERGING '        EMIGRA             ENUMERA            ERADIAT
                    ERUPT              EVAL               EVAPO              EVISCERAT          EVOC
                    EVOK               END CLUD
EM          1                          ENCLOS,PUT INTO OR ON;GIVE THE QUALITY,AG
                    EMBALM             EMBANK             EMBATTL            EMBED              EMBITTER
                    EMBLAZ             EMBOD              EMBOLDEN           EMBOSOM            EMBOW
                    EMBOWEL            'EMBRACE '         EMBRITTL           EMBROIDER          EMBUS
                    EMPANEL            EMPLAC             'EMPLOY '          EMPOISON           EMPOWER
                    END CLUD
EN          1                          IN, OR VB FORM. OR TRANSITIVE
                    ENABL              ENACT              ENAMOR             ENCAG              ENCAMP
                    ENCAPSUL           ENCAS              ENCHAIN            ENCLASP            ENCLOS
                    ENCOD              ENCOMPASS          ENCOURAG           ENCRUST            ENCYST
                    ENDANGER           ENDEAR             ENDUR              ENFAC              ENFEEBL
                    ENFOLD             ENFORC             ENFRANCHIS         ENGAG              ENGENDER
                    ENGIRD             ENGORG             ENGRAIN            ENGRAV             ENHEARTEN
                    ENJOIN             ENJOY              ENKINDL            ENLAC              ENLARG
                    ENLIGHT            ENLIST             ENLIV              ENMESH             ENRAPT
                    ENREGISTER         ENRICH             ENROLL             ENSAMPL            ENSHRIN
                    ENSHROUD           ENSLAV             ENSNAR             ENSUR              ENTANGL
                    ENTHRON            ENTITL             ENTOMB             ENTRAIN            ENTRANC
```

```
PREFIX   KEY    CLUDLIST

                ENTRAP          ENTRENCH        ENTRUST         ENTWI           ENVISAG
                ENVISION        ENWRAP          ENWREATH        END CLUD

EPI      C                      (AT,BEFORE,AFTER)
                'EPIC '         EPICURE         EPIDERM         EPIGENE         EPIGRA
                EPILEP          EPILOG          EPIPHENOMEN     EPISCO          EPISO
                EPISTLE         EPITAPH         EPITHELI        EPITHET         EPITOME
                END CLUD

ERE      1                      (BEFORE,-ARCHAIC-)
                ERELONG         ERENOW          EREWHILE        END CLUD

EX       1                      EX
                EX_             EXCENTRIC       EXCHANG         EXCURRENT       EXCURS
                EXPORT          END CLUD

EXTRA    C                      OUTSIDE,ADDITIONAL,MORE THAN USUAL,SUPERI
                EXTRACT         EXTRAD          EXTRAMURAL      EXTRANEOUS      EXTRAPOLAT
                EXTRAVAGAN      EXTRAVER        END CLUD

FARM     0                      (FARM)
                FARME           FARMI           END CLUD

FAT      1                      (FAT)
                FAT-FACED       FATFREE         FATHEAD         END CLUD

FOR      1            AWAY,OFF,EXTREMELY,WRONGLY,NEGATIV OR PRIVATIV FOR
                FORBAD (?)
                'FORBEAR ' (?)
                'FORBID ' (?)
                FORBOR (?,,E,NE)                                FORDO
                FORFEND (ARCHAIC)
                FORGIV (?)                                      'FORGO '
                FORSAKE (?)
                FORSOO (?,K,TH)
                FORSPENT (?,ARCHAIC)
                FORSWEAR
                FORSWOR (E)                                     FORWENT         END CLUD

FORE     0            BEFORE__,FRONT,SUPERIOR
                FORE_AND
                FOREBODING (?)                                  'FORECAST '     'FORECASTS '
                FOREDID         'FOREDO '       FOREDOING       FOREDONE        'FOREGO '
                FOREIGN         FORENSIC        'FOREST '       'FORESTED '     FORESTATION
                FORESTER        FORESTRY        FOREVER         END CLUD

GEO      1            THE EARTH
                GEOCENTRIC      GEOGRAPHIC      GEOPHYSIC       END CLUD

GOAL     1                      (GOAL)
                GOALK           GOALTEND        END CLUD

GUIDE    0                      (GUIDE)
                GUIDED          END CLUD

HAIR     0                      (HAIR)
                'HAIR '         HAIRBRA         HAIRDO          HAIRLES         HAIRSPLIT
                HAIRY           END CLUD
```

```
PREFIX      KEY    CLUDLIST

HALF        1                           HALF
                   HALF_AND_HALF        HALF_BLOOD           HALF_HEARTED         HALF_TRACK           HALFWAY
                   END CLUD

HEMI        1                           HALF
                   HEMISPHER            END CLUD

HETERO      1                           DIFFERENT,OTHER
                   HETEROCHROM          HETEROSEX            END CLUD

HEXA        1                           SIX
                   HEXAMETER            HEXANGULAR           HEXASYLLABL          END CLUD

HIND        1                           REAR,PAST
                   HINDQUARTER          HINDSIGHT            END CLUD

HOMO        1                           SAME
                   HOMOCENTRIC          HOMOCHRO             HOMOSEX              END CLUD

HUMAN       1                           (HUMAN)
                   HUMAN-INTER          HUMANHEART           HUMANHOOD            HUMANKIND            HUMANMIND
                   END CLUD

HYDRO       0               WATER,HYDROGEN
                   HYDROCHLORIC         HYDROGEN             HYDROGRAPHY          HYDROLOGY            HYDROLY
                   HYDROPHOB            HYDROUS              END CLUD

HYLO        1                           WOOD,MATTER
                   HYLOTHEIS            END CLUD

HYPER       0                           OVER,(SEM.DIF.EXCESS)
                   HYPERBO              HYPERURBAN           END CLUD

HYPNO       1               SLEEP,HYPNOSIS
                   HYPNOANALY (SIS)
                   HYPNOG (ENESIS,RAPH)
                   END CLUD                                                                        HYPNOTHERAP

HYPO        0                           UNDER,LESS,LOW
                   HYPOTHESIS           HYPOTHETC            END CLUD

ICONO       1                           IMAGE,LIKENESS
                   ICONOGRAPH           END CLUD

IL          1               VAR.OF IN,NOT,VB.FORM .
                   ILLAUDAB             ILLEGAL
                   ILLEGI (BLE,TIMATE)                                                             ILLIBERAL
                   ILLICIT              ILLIMIT
                   ILLITERA (?A)                                                                   ILLOGIC
                   ILLUMIN (A TEST)                          END CLUD

IM          1               VAR.OF IN,NOT,ETC.
                   IMBALANC             IMBALM               IMBARK               IMBITTER
                   IMBO (DY,SOM,WER?,)
                                        IMBRANGL             IMBRUT               IMMACULAT            IMMERG
                   IMMESH               IMMIGRA              IMMINGL              IMMISCIB             IMMIX
                   IMMOBIL
                   IMMOD (ERATE,EST)
```

```
PREFIX     KEY      CLUDLIST

                    IMMOR (TAL,AL)
                    IMMOTI (LE)                          IMMOV
                    IMMUSIC (AL)
                    IMMUTA (BLE)
                    IMPARADI (SE)
                    IMPARIT (Y)
                    IMPARK (COMMON ENOUGH?)                                   IMPARTIB            IMPASSAB
                    IMPASSION
                    IMPATIEN (CE,T)                                           IMPEND
                    IMPENET (RABLE)
                    IMPENIT (ENT)
                    IMPERC (EPTIBLE,IPIENT)
                    IMPERF (ECT,ORATE)                   IMPERIL              IMPERISHAB
                    IMPERM (ANENT,EABLE,ISSIBLE)
                    IMPERSON (ATE)
                    IMPERT (INENT_1M,URB)
                    IMPLAC (ABLE,ENTAL)                  IMPLANT
                    IMPLAUS (IBLE)
                    IMPOLI (CY,TE)                       IMPOND
                    IMPOSS (IBLE)
                    IMPOT               IMPOVER          IMPOWER
                    IMPRACT (ICAL)
                    IMPREC (ISE)
                    IMPREGNAB (LE,)                                                               IMPRINT
                    IMPRISON
                    IMPROB (ABLE,ITY)
                    IMPROMPT (U)
                    IMPROPER
                    IMPROPRIET (Y)
                    IMPROVIDEN (T,CE)
                    IMPRUDEN (T,CE)
                    IMPUISSAN (T,CE)                                                              IMPUS
                    IMPUTRESCIB         END CLUD

IN         0             UN,NOT,IN(TO),VB.FORMATIVE & TRANSITIVE
                    IN_AND              INADVERT         INAM                 INANE               INANI
                    INARM               INASMUCH         INAUGURA             INCANDESC           INCANT
                    INCAR (CERATE,DINATE,NALIZE)
                    INCEN (DIARY,SE,TIVE)
                    INCEP (TION,TIVE,T)
                    INCES (T,SANT)                                            INCH
                    INCID (ENCE,ENT)
                    INCIN (ERATE)
                    INCIP (IENT,IT)
                    INCIS (E,ION,ISIVE,SOR)
                    INCIT (E)
                    INCLE (MENT)
                    INCLI (NE,)
                    INCLU (DE)
                    INCOG (TOO COMPLEX - DEBUG BY RUNNING)   INCREAS          INCREM
                    INCRESC (ENT)       INCRET           INCRIMIN
                    INCUB (ATE,US)
                    INCULCAT            'INCULPATE '
                    INCULT (ARCHAIC,UNCULTIVATED)                                                 INCUMBRA
                    INCUMBEN            'INCUR '         INCURRE              INCURS
                    INDEMNI (FICATION,TY)
                                        INDENT           INDEX                INDIA               INDIC
                    INDIGEN             INDIGNA          INDIGO               INDIT               INDIVIDUA
```

```
PREFIX    KEY     CLUDLIST

                  INDO_                   INDOCH
                  INDOL (E,ENT)
                  INDU (RATE,STRY)
                  INFBRI (ETY,ATE)                                  INERT             INFAM
                  INFANT                  INFECT            INFER   INFEST            INFIDEL
                  INFIRMAR (Y)
                  INFIRMI (TY)                                                        INFIX
                  INFLAT                  INFLECT           INFLICT      'INFORM '    'INFORMANT '
                  'INFORMAT ' (ION,ORY,IVE)                 'INFORMED '  'INFORMER '  'INFORMING '
                  INFRA                   INGEST            INGO         INGRATIAT    INGRED
                  INHABIT                 INHAL
                  INHE (RIT,SION)                                                     INHIBIT
                  INI (MICAL,MITABLE,TIAL)                  INJECT       INJUN        INJUR
                  INK                     INNE              INNO
                  INOC (ULATE)
                  INQUIR                  INQUIS            INROAD       INSCR        INSECT
                  INSERT                  INSIGNE           INSIGNIA     INSINUAT     INSIP
                  INSIST                  INSOFAR
                  INSOLA (TE,)
                  INSOLE (NT,E)           INSOMNIA          INSOMUCH
                  INSPECT (OR,ORATE)
                  INSPIR (IT,RE,)
                  INSTAL (L,LATION)
                  INSTAN (CE,CY,T)        INSTI
                  INSTRU (CT,MENTAL)                                              INSUL
                  INSURA (NCE)                              INSURE
                  INSURG (ENT)
                  INSURRECT               INTAK
                  INTEG (ER,RAL,RATE)
                  INTELL (ECT,IGENCE)
                  INTEN (D,DANT,SE,T)
                  INTER (SHOULD WORK EXCPT 'MINABLE')
                  INTESTIN                INTHRALL
                  INTIM (ATE)                                                     INTIN
                  'INTO '
                  INTOXICA (NT,ATE)
                  INTRA_ (INTRA SHOULD BE RUN-DEBUGGED)
                  INTRAM (VAR.)
                  INTRAS (PINAL,TATE)
                  INTRAV (VAR.)           INTREPID
                  INTRI (CATE,GANT,GUE,NSIC)
                  INTRO (VAR.)            INTRUD            INTRUSI      INTUIT
                  INUNDA (TE)
                  INUR (E,N)                                             INVAD
                  INVAS (ION)             INVECT
                  INVEI (GH,GLE)                                                      INVENT
                  INVERSE                 'INVERT           'INVERTED '  INVERTI
                  INVET (ERATE)
                  INVID (IOUS)
                  INVIG (ILATE,ORATE)
                  INVIT                   INVOC             INVOIC       INVOK        INVOLV
                  END CLUD

INFRA     0                 BELOW
                  INFRACT                 INFRANGIB         END CLUD

INTER     0                 AMONG,BETWEEN,MUTUALLY,DURING,ETC.
                  INTERCED                INTERCEPT
```

```
PREFIX      KEY     CLUDLIST

                    INTERCESS (ION)                                                                       'INTERCOM '
                    INTERDICT             INTEREST              INTERFER              'INTERIS '          INTERIOR
                    INTERJACEN (CE,T)                           INTERJECT             INTERLUDE           INTERMEDIAT
                    INTERMENT             INTERMINAB            INTERMISSI            INTERMIT            INTERMURAL
                    'INTERN '             INTERNAL              INTERNED              INTERNEE            INTERNING
                    INTERNIST             INTERNMENT
                    INTERPEL (LANT)                                                                       'INTERPOL
                    INTERPOLAT            INTERPRET
                    INTERR (OGATE,UPT)                                                                    INTERSECT
                    INTERSPERS
                    INTERSTIC (E)
                    INTERSTIT (IAL)
                    'INTERVAL '           'INTERVALS '          INTERVEN              INTERVIEW           INTERVOLV
                    END CLUD

IR          0                IN,NOT,VB.FORMATIVE & TRANSITIVE
                    IRA                   IRE                   'IRISH '              IRO                 IRREMEAB
                    IRRIGA
                    IRRIGUOUS (ARCHAIC,WELL_WATERED)                                  IRRITA

KEY         1                KEY(LOCK),CENTRAL IMPORTANCE
                    KEYHO                 KEYMAN                KEYNOTE               KEYPUNC             KEYSMI
                    KEYSTONE              KEYSTROK              KEYWORD               END CLUD

LITHO       1                             STONE
                    LITHOGRAPH            LITHOPRINT            LITHOSPHER            END CLUD

MACRO       0                LARGE,LONG.EXCESSIVE,NO WORDS-SOME ARE RARE
                    END CLUD

MAL         1                BAD,WRONG,ILL, FR.
                    MALADAPT              MALADJUST             MALADMINIS            MALAPPORTION        MALCONTENT
                    MALF (EASANCE,ORM,UNCTION)                  END CLUD

META        0                             AFTER,AWAY,BEYOND,BEHIND
                    METABOLI              METAL                 METAMER               METAMOR             METAPHOR
                    END CLUD

MICRO       0                SMALL,ENLARGING SOMETHING SMALL
                    'MICROBE '            END CLUD

MID         1                MIDDLE,BETWEEN
                    MIDA (IR,FTERNOON)
                    MIDB (RAIN,AND)
                    MIDC (OURSE)          MIDDAY
                    MIDL (AND,EG,INE)
                    MIDMO (ST,ON)
                    MIDN (IGHT,OON)                                                   MIDPOINT            MIDRASH
                    MIDSECTION            'MIDSHIP '            'MIDSHIPS '           MIDSTREAM           MIDSUMMER
                    MIDT (OWN,ERM)
                    MIDW (ATCH,AY,EEK,EST,IFE,INTER)                                                      END CLUD

MIS         0                ILL,MISTAKEN,WRONG
                    MISCE (GENATION,LLANY)
                    MISCHA (NCE,NTER)
                    MISCHIE (F)           MISCI                 MISCREAN              'MISE '             MISER
                    MISGIV                MISHAP                MISHMASH              MISNOMER
                    MISO (='HATE' PREFIX)
```

```
PREFIX      KEY     CLUDLIST

                    MISPRIS (E,ION)
                    MISSI (LE,ON)                               MISSY               'MISS '             'MISSED '
                    MISTER            MISTLE                    MISTOOK             'MIST '             MISTAK
                    END CLUD                                                        MISTRESS            MISTY

MON         1                ALONE,SINGLE,ONE,VAR. OF 'MONO'
                    MONAURAL          END CLUD

MONO        C                ALONE,SINGLE,ONE,
                    MONOLITH (IC)
                    MONOPOL  (Y)                                                                        END CLUD

MULTI       0                MANY
                    MULTIFARIOUS      MULTIPID
                    MULTIPAR (A,OUS)                                                                    MULTIPLE
                    MULTIPLI          MULTIPLY                  MULTITUD            END CLUD

NEO         0                         NEW,RECENT
                    NEOLITH           NEOLOG                    'NEON '             NEOPHYT             END CLUD

NO          1                NO
                    NO_
                    NOBODY  (?)                                                     NOWAY               NOWHERE
                    END CLUD

NON         0                NOT,'LACKING_',NOT NECESSAIRILY 'REVERSE'
                    'NONAGE '         'NONCE '
                    NONCHALAN (T,CE)                                                                    'NONCOM '
                    'NONDESCRIPT '    'NONE '                   NONESUCH            NONPAREIL           NONPLUS
                    NONSUCH           END CLUD

OB          1                         TOWARD,ON,OVER,AGAINST
                    OBLIGAT           OBLONG                    OBNOX               OBVER               END CLUD

OFF         1                OFF
                    OFF_              OFFBEAT                   OFFCAST             OFFHAND             OFFPRINT
                    OFFS (COURING,CREEN,SET)                    OFFTAK              END CLUD

OUT         0                OUT+TRANS.VB.GOING BEYOND,SURPASSING,OUTDOING
                    OUT_OF            OUTAG                     OUTER               OUTFIT              OUTLIER
                    OUTRAG (E,EOUS)                             'OUTSET '           OUTSIDE             OUTWARD
                    END CLUD

OVER        0                OVER A LIMIT
                    'OVER '           OVERLAP                   OVERSEER            'OVERT '            END CLUD

PAN         1                         ALL,GENERAL
                    PANSOPHISM        PANTHEISM                 PANTROPIC           END CLUD

PARA        1                PARACHUT,GUARD AGAINST,BESID,NEAR,AMISS,+IMP.ALTER
                    PARABOMB          PARACHUT                  PARAGLIDER          'PARAMEDIC '        PARAMAGNET
                    PARAMILIT         PARAPHRAS                 PARAPSYCHO          PARARESCU           PARASOL
                    PARATROOP         END CLUD

PAY         1                TO PAY ETC.
                    PAYDAY            PAYLOAD                   PAYMASTER           END CLUD

PER         0                THROUGH,UTTERLY,VERY,THOROUGHLY
```

```
PREFIX      KEY     CLUDLIST

                    PERAMBULAT
                    PERDUR (E,ABLE)                                             'PERFECT '          PERFERVID
                    'PERHAPS '          PERMUTAT            'PERSON '
                    PERSUA (DE,SION)
                    END CLUD

PERI        1                 ABOUT,AROUND,BEYOND
                    PERISCOP            END CLUD

POLY        1                 MULTIPLE,MUCH,MANY
                    POLYANG (ULAR)                          POLYCHROM           POLYETHNIC          POLYGENE
                    POLYGRAPH           POLYPHON            POLYSYLLAB          POLYTECHNIC         POLYTHEIS
                    POLYTON             POLYTYP             END CLUD

POST        (                 BEHIND,AFTER,MAIL
                    POSTAGE             POSTAL              POSTER              POSTIC              POSTING
                    POSTPONE
                    POSTU (LATE,RE)                                             END CLUD

PRE         (                 BEFORE,PRIOR TO,EARLY,IN FRONT OF
                    PREACH              PREAMBL             PRECARI
                    PRECAT (ORY)
                    PRECE (DE,PT,DENT)
                    PRECI (PICE,SE)
                    PRECL (UDE)
                    PRECOG (NITION?)                                            PRECUR              PREDACIOUS
                    PREDATOR            PREDECESS
                    PREDIC (T,AMENT,ABLE)                                                           PREEMPT
                    'PREEN '            'PREFAB '           PREFACE             PREFAT
                    PREFE (CT,R,RENCE)  PREFIX
                    PREGNAN (T,CY)                                                                  PREHENS
                    PREJUDIC
                    PRELA (TE,CY,TURE)                                          PRELIMIN
                    PRELU (DE)
                    PREMIE (RE)                                                 PREMISE             PREMIUM
                    PREMON              PRENTICE            'PREP '             PREPARA             'PREPARE '
                    PREPARED            PREPENSE            PREPOSITION         PREPOSSES           PREPOSTEROUS
                    PREPUCE             PREROG              PRESBYTER           PRESCIND            PRESCRIBE
                    PRESCRIPT           PRESENC             PRESENT             PRESERV             PRESID
                    PRESS               'PREST '            PRESTI
                    PRESUM (PTION)
                    PRETEN (CE,D,DER,SE,SION,TIOUS)         PRETER              PRETT
                    PREVA (IL,RICATE)
                    PREVEN (T,IENT)                                             PREVIOUS            PREXY
                    PREY                END CLUD

PRETER      (                 BEYOND,MORE THAN,BY,PAST
                    PRETERI             PRETERMIT           END CLUD

PRO         (                 FOR (A CAUSE,ETC.)
                    PRO_SAT
                    PROBAB (LE)
                    PROBAT (E)
                    PROBE               PROBITY             PROBLEM
                    PROCED (URE)
                    PROCEED             PROCESS             PROCLAIM
                    PROCLAM (ATION)
                    PROCLIVITY          PROCRAST            PROCREANT
```

```
PREFIX      KEY     CLUDLIST

                    PROCTO (PREFIX,ANUS,RECTUM)
                    PROCUR              'PROD '             PRODDE
                    PRODIG (Y,AL,IOUS)
                    PRODU (CE,CT)                           'PROF '             PROFAN              PROFESS
                    PROFF
                    PROFICI (ENT,ENCY)
                    PROFIL (E)
                    PROFIT              PROFLIG             PROFOUND            PROFU               PROGEN
                    PROGN (OSE,OSTIC)                       PROGRAM             PROGRESS            PROHIB
                    PROJECT             PROLE
                    PROLIF (IC,ERATE)                                                               PROLIX
                    PROLOG
                    PROLONGAT (E,ION)                                           PROMENAD
                    PROMINEN (CE,T)
                    PROMIS (E,SORY)
                    PROMON (? SP.DICT.)
                    PROMOT              PROMPT
                    PROMULG (ATE,E)                                                                 'PRONE
                    PRONG               PRONOUNC            PRONTO              PRONUN              PROOF
                    PROPAG (ABLE,ANDA,AGATE)                PROPEL              PROPEN              PROPERTY
                    PROPH (ESY,ET,Y,YLACTIC)
                    PROPI (NE,NQUITY,TIATORY)                                                       PROPJET
                    PROPON (E,ENT)
                    PROPOS (E,ITION)                                                                PROPOUND
                    PROPRIET
                    PROPRIO (PREFIX)
                    PROPULS (ION)
                    PROSAI (C,SM)
                    PRORAT (?)                                                                      PROSCRIB
                    PROSF
                    PROSOD (ENSIC,Y,EST)                                        PROSPECT            PROSPER
                    PROSTHE (SIS,TICS)                      PROSTITUT           PROSTRAT
                    PROTAG (ONIST,AGORAS)
                                        PROTEAN             PROTECT
                    PROTEG (E)
                    PROTEIN             PROTEST
                    PROTO (PREFIX)
                    PROTRACT (?,SUGGEST RUN_DEBUG)
                                        PROTRU
                    PROTU (BERANCE)                                                                 'PROUD '
                    PROVINC             END CLUD

PROTO       1                 FIRST,FOREMOST,EARLIST FORM OF,MAYB SHOULD 0 IF SCI
                    PROTOHUMAN          PROTOLANGUAGE       PROTOLINGU          PROTOPLASM          PROTOTYP
                    END CLUD

PSEUDO      0                 FALSE,PRETENDED
                    PSEUDOMORPH
                    PSEUDONYM (ITY,OUS)                                         END CLUD

QUASI       0                       RESEMBLING,SEEMING
                    END CLUD

RE          0                 BACKWARD,AGAIN
                    REACH               REACTA              'READ '             READER              READI
                    READY               'REAL '             REALIS              REALIT              REALIZ
                    REALLY              REALM               REALPOLITIK         REALTY              'REAM '
                    'REAP '             REAPER              REAPI               'REAR '             REARMOST
```

```
PREFIX     KEY     CLUDLIST

                   REARWARD            REASON              REBATE              REBEL               REBUFF
                   REBUK               'REBUT '            REBUTTAL
                   RECALCITRA (NT,TE)
                   RECANT (?RUN DE_BUG)                    RECED
                   RECEI (PT,VE)
                   RECENSION           'RECENT '
                   RECEPT (ACLE,IBLE,ION,IVE,OR)                                                   RECESS
                   RECIDIV
                   RECIP (IENT,ROCAL)                                          RECIS               RECIT
                   RECK                RECLAM              RECLIN              RECLUS
                   RECOGNI (SE,TION,ZANCE,ZOR)
                   RECOIL (ONLY 'REWIND' MEANING SHOULD BE CHOPPED)
                   RECOLLECT (ONLY 'REGATHER' MEANING SHOULD BE CHOPPED)
                   RECOMPENSE          RECOMPENSI          'RECON '            RECONCIL            'RECONDITE '
                   RECONNAISANCE       RECONNOIT           RECORD              RECOUP              RECOURS
                   RECOVER             RECREANT            RECRIMINAT          RECRUIT
                   RECT (VAR.)
                   RECUM (BENT)
                   RECUP (ERATE)
                   'RECUR '            RECURR
                   RECURS (ION)                                                                    'RED '
                   RED_ (VAR.)
                   REDB (VAR.)
                   REDD (VAR.)         REDEEM              REDEMPT             REDIN               REDN
                   REDOLEN (T,CE)                          REDOUBT             REDRESS
                   REDS (KIN)
                   REDUC (E)                                                   'REED '             REEF
                   REEK
                   'REEL ' (? ER,ED,)                                          REELING             REEVE
                   REFECT              REFER               REFINE              REFLAT              REFLECT
                   REFLEX
                   REFORMATORY (? REFORM?)                                     REFORMIST           REFRACT
                   REFRAIN
                   REFRI (GERATOR,NGENT)                                       REFUG               REFUS
                   REFUT               REGAL               REGARD              REGENCY             REGENT
                   REGICID             REGIM               REGION              REGIST              REGRESS
                   REGRET              REGULA              REGURGITAT          REHASH              REHEARS
                   REICH               REIGN               REIMBURS            'REIN '
                   'REINS ' (ARCHAIC _ THE KIDNEYS)
                                       REITERANT           REJECT              REJOIC              REJOINDER
                   KEYBO               REJUVEN
                   RELAT (IVE)                                                                     RELAX
                   RELAY               RELEAS              RELEG               RELENT              RELEVANT
                   RELIAB              RELIAN              'RELIC '            'RELICS '
                   RELIE (F,VE)
                   RELIGIO (N)                                                 RELIQUE             RELISH
                   RELUC               REMAIN              REMAND              REMARK              REMEDIAB
                   REMEDIAL            REMEDILESS          REMEDY              REMEMB              REMIND
                   REMINISC            REMISS              REMIT
                   REMNAN (T)
                   REMONSTRA (NT,CE,TE)
                   REMORS (E IS INTENDED)                                                          REMOTE
                   REMOVAL             REMUNERAT           REND                RENEGAD             RENEGE
                   RENEWAL             RENOUNC             RENOVAT             RENOWN              RENT
                   RENUNC              REPAIR              REPARA              REPARTE             REPAST
                   REPEA (L,T)                             REPEL
                   REPERTO (IRE,RY)
                   REPETIT             REPLENISH           REPLET              REPLI               REPLICA
```

```
PREFIX    KEY     CLUDLIST

                  REPLY                  REPORT                 REPOSAL                RL[illegible]          REPOSITORY
                  REPREHEN               REPRESENT              REPRESS                REPRIEV                REPRIMAND
                  REPRIS                 REPROACH               REPROBAT               REPROOF                REPROV
                  REPTIL                 REPUBLIC               REPUGN                 REPULS                 REPUT
                  REQUEST                REQUIF                 REQUIF                 REQUIS
                  REREMOUSE (ARCHAIC)
                  RESCI (ND,SSION)                                                     RESCU                  RESEARCH
                  RESEMBL                RESENT                 RESERV                 RESID                  RESIGN
                  RESILE                 RESILI                 RESIN                  RESIST                 RESOLU
                  RESOLV
                  RESONA (NT,NCE,TE)                                                   RESORT                 RESOURC
                  RESPECT                RESPIR                 RESPIT                 RESPLEND               RESPON
                  'REST '                'RESTED '              RESTAUR                RESTFUL                RESTITUT
                  RESTIVE                RESTLESS
                  RESTORAT (IVE,ION)                                                                          RESTRAIN
                  RESUM                  RESURG                 RESURR                 RESUSC
                  RETAI (L,N)
                  RETAL (IATE)                                                         RETARD                 RETCH
                  RETENTI (ON,VE)
                  RETIC (LE,ULAR)                                                                             RETICEN
                  RETIN (A,UE,VAR.SCI.)                         RETIR                  'RETORT '              RETRACT
                  RETREAT                RETRIBUT               RETRIEV                RETRO                  RETRU
                  RETURN                 REVEAL                 REVEL                  REVEN                  REVER
                  REVIS                  REVIV                  REVOC                  REVOK
                  REVOL (T,UTION)        REVULS                 REWARD
                  REX (? NECESS?)
                  REY (? NECESS?)                               END CLUD

RETRO     0                              BACKWARD,BEHIND
                  RETROCED               RETROGRADE             RETROGRESS             RETROSPECT             RETROVERSION
                  END CLUD

SELF      1                 COMB.FORM OF 'SELF'
                  SELF_                  SELFSAME               END CLUD

SEMI      0                 HALF
                  SEMINA                 SEMINI                 SEMITIC                END CLUD

SIDE      0                 SIDE
                  SIDEBURN               SIDEKICK               SIDER                  'SIDE '                'SIDES '
                  SIDESPLIT              SIDEWINDER             END CLUD

STEP      0                 STEP
                  STEPH
                  STEPP (E,ED,ING)                                                     'STEPS '               STEPWISE
                  END CLUD

SUB       0                 BELOW,SLIGHTLY,(NOTION OF ASSISTANCE)
                  SUBALTERN              SUBDUCT                SUBDUE                 SUBER                  SUBJECT
                  SUBJUNCT (IVE,ION)                            SUBLIMAT               SUBLIME                SUBMERG
                  SUBMERS                SUBMISS                SUBMIT                 SUBORN                 SUBSCRIB
                  SUBSCRIPT              SUBSID                 SUBSIST                SUBSTANC               SUBSTANT
                  SUBSTITUT (UENT,UTE,UTIVE)
                  SUBSUM (E,PTION)
                  SUBTER                                                                                      SUBTEND
                  SUBTILE (ARCHAIC SUBTLE)                                             SUBTILI
                  SUBTILLY (ARCHAIC,SUBTLETY)
                                         SUBTLE
```

```
PREFIX     KEY     CLUDLIST

                   SUBTRA (CT,HEND)                                                        'SUBURB '
                   'SUBURBIA '         'SUBURBS '
                   SUBVE (NE,NTION,RSION,RT)                                               END CLUD

SUBTER     1                 UNDER,BELOW
                   SUBTERNATURAL       END CLUD

SUPER      0                 ABOVE,BEYOND,TO AN ESPEC. HIGH DEGREE
                   SUPER_DUPER         SUPERABLE           SUPERANNUAT         'SUPERB '           SUPERCARGO
                   SUPERCIL            SUPERFGO            SUPERETTE           SUPERFACIECT        SUPERFIC
                   SUPERFLUITY         SUPERFLUOUS         SUPERFUS            SUPERINTEND         SUPERIOR
                   SUPERLATIV          SUPERNAL            SUPERORDINAT        'SUPERPOSE '        'SUPERPOSED '
                   'SUPERPOSING '      SUPERPOSITION       SUPERSCRIBE         SUPERSCRIPTION      SUPERSED
                   SUPERSESSION        SUPERSTIT           SUPERVENE           SUPERVIS            END CLUD

SUPRA      0                 VAR.OF 'SUPER' EMPHASIZING POSITION
                   END CLUD

SUR        1                 VAR.OF 'SUPER',VAR.OF 'SUB'
                   SURCEAS (ARCHAIC DESIST,SUR(1)+CEASE)   SURCHARG            SURCOAT
                   SURFACE (?)         SURMOUNT            SURNAM              SURPASS             SURPLUS
                   SURPRINT            SURREAL             SURROUND            SURTAX              END CLUD

SYM        1                 WITH,TOGETHER
                   SYMMETRIC (?)                           END CLUD

SYN        1                 WITH,TOGETHER,IN ASSOC.(WITH)
                   SYNAESTHESIA        SYNECOLOGY          SYNESTHESIA         SYNGENESIS          END CLUD

SYNCHRO    0                           SYNCHRONOUS
                   SYNCHRONAL          SYNCHRONISE         SYNCHRONISM         SYNCHRONIZ          SYNCHRONOUS
                   END CLUD

TAX        1                 ORDERING,DIRECTION,TAX
                   TAX_                TAXGATHER
                   TAXP (AID,AYER)                                                         END CLUD

TAXI       1                 TAXI(CAB),VAR.OF TAXO
                   TAXIMETER           TAXIPLANE           TAXIWAY             END CLUD

TETRA      0                           FOUR
                   TETRAD              TETRAHED            TETRAMER            END CLUD

THOROUGH   0                 THOROUGH, THROUGH
                   THOROUGHFARE        END CLUD

THROUGH    1                 THROUGH
                   THROUGHPUT          THROUGHWAY          END CLUD

TRANS      0                 ACCROSS,BEYOND,THROUGH
                   TRANSCEIVER         TRANSCEND           TRANSCIEN           TRANSCRIPTION       TRANSDUCE
                   TRANSECT            TRANSFER            TRANSFORMER         TRANSGRESS          TRANSIST
                   TRANSIT             TRANSLAT            TRANSLITERAT        TRANSMISS           TRANSMIT
                   TRANSOM             TRANSPAREN          TRANSPIR            TRANSPORT           TRANSPOS
                   TRANSUD             TRANSVERS           END CLUD

TRI        1                 THREE
                   TRI_STATE (PUTPLACE)                    TRIANGLE            TRICHROMAT          TRICOLOR
```

```
PREFIX     KEY    CLUDLIST

                  TRICYCL              TRIFORM              TRILINGUAL           TRIMETALLI           TRIMONTHLY
                  TRIMOTOR             TRINITRO             TRIPEDAL             TRIPLANE
                  TRISYLLAB (LE)
                  TRIWEEK (LY)                                                   END CLUD
TROPO       1             TURN,TURNING
                  TROPOSPHER           END CLUD
ULTRA       0                          BEYOND USUAL,EXCESSIVE
                  ULTRAISM             END CLUD
UN          0             UN,NOT,LACKING IN,ONE
                  UNANIM
                  UNCANNY ('CANNY' HAS ARCHAIC MEANING 'SUPERNATURAL)
                  UNCHANY (?)
                  'UNCLE '             UNCT                 UNDER                UNDIES
                  UNDULA (NT,TE)       UNGUENT
                  UNGUL (AR)                                                                          UNIAXIAL
                  UNIC (ORN,CYCLE)
                  UNID (IRECTIONAL)                                                                   UNIF
                  UNIL (INGUAL,OBED,ATERAL)                 UNION                END CLUD
UNDER       0             ?UNDER,ONE?
                  UNDER_THE (PUTPLACE)                      UNDERLING            UNDERNEATH           UNDEROGATORY
                  UNDERSTAND           UNDERTAK             UNDERTOOK            END CLUD
UNI         1             ONE
                  UNIAXIAL             UNICAMERAL           UNICYCL              UNIDIRECT
                  UNIFORM (?WORD IN ITSELF?)
                                       UNILATERAL
                  UNILING (UAL)
                  UNILO (BED,CULAR)
                  UNIP (ERSONAL,LANAR,OLAR,OTENT)                                UNISEX               'UNIVERSE '
                  END CLUD
UP          0             UP
                  UP_AND               UP_TO                UPBRAID              UPBRINGING           UPHEAVAL
                  UPHOLSTER            UPON                 UPP                  UPSET                UPSHOT
                  UPWARD               END CLUD
VICE        0                          DEPUTY
                  VICELES              VICEN                END CLUD
WELL        0                          GOOD
                  'WELL '              WELL-FAVOR           WELL-FIX             WELL-HEEL            WELL-OFF
                  WELL-OIL             WELL-SPRING          WELL-TO-DO           WELL-TURN            WELLAWAY
                  'WELLS '             END CLUD
WITH        1            COMBINING FORM OF WITH,SEPARATIVE OR OPPOSING FORC
                  'WITHDRAW '          'WITHDRAWING '       WITHDREW             WITHOLD              WITHIN
                  WITHOUT (?)
                  WITHSTAND (?)                                                                       END CLUD
XYLO        1                          WOOD
                  XYLOGRAPH            XYLOPHON             END CLUD
YESTER      0                          PRECEDING
                  END CLUD
```

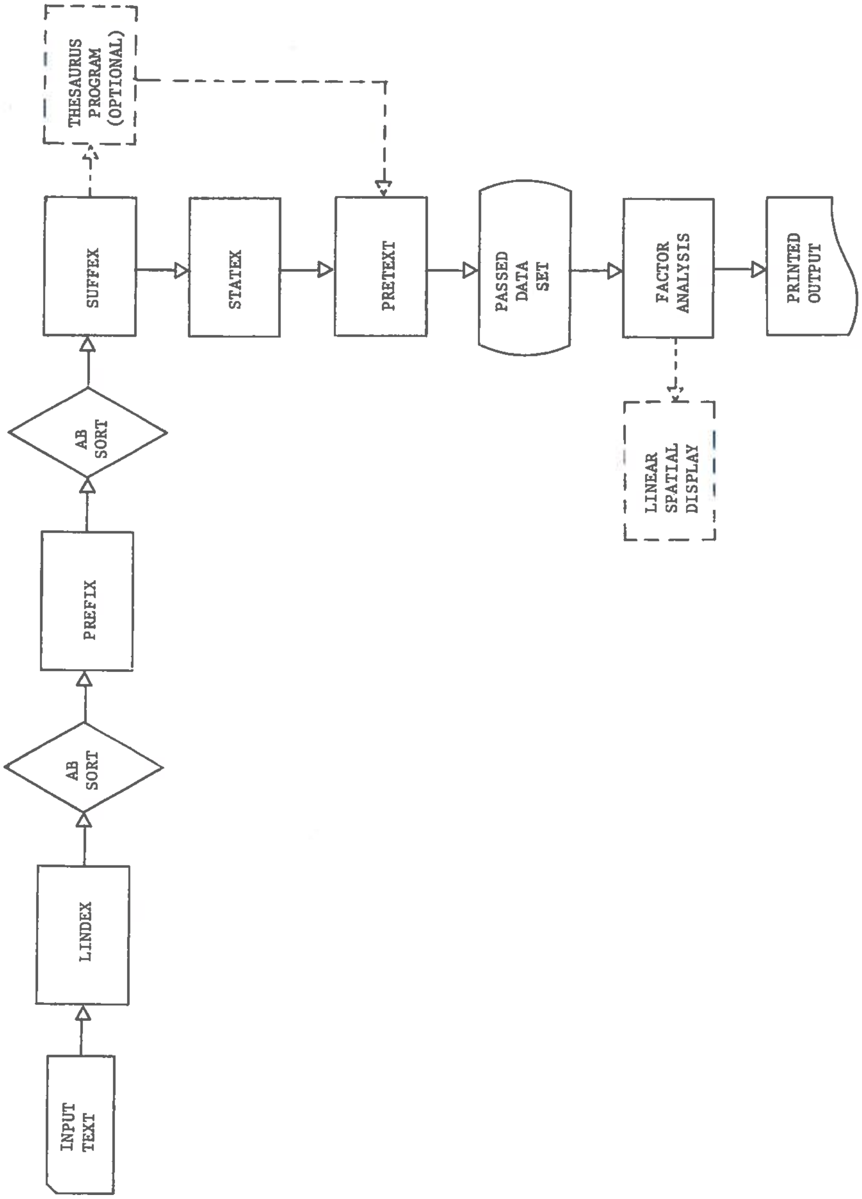
```
PREFIX     KEY     CLUDLIST

ZYGO       1                           SCI. UNION,CONNECT
                   ZYGOGENESIS         ZYGOSPORE           END CLUD
```

APPENDIX G

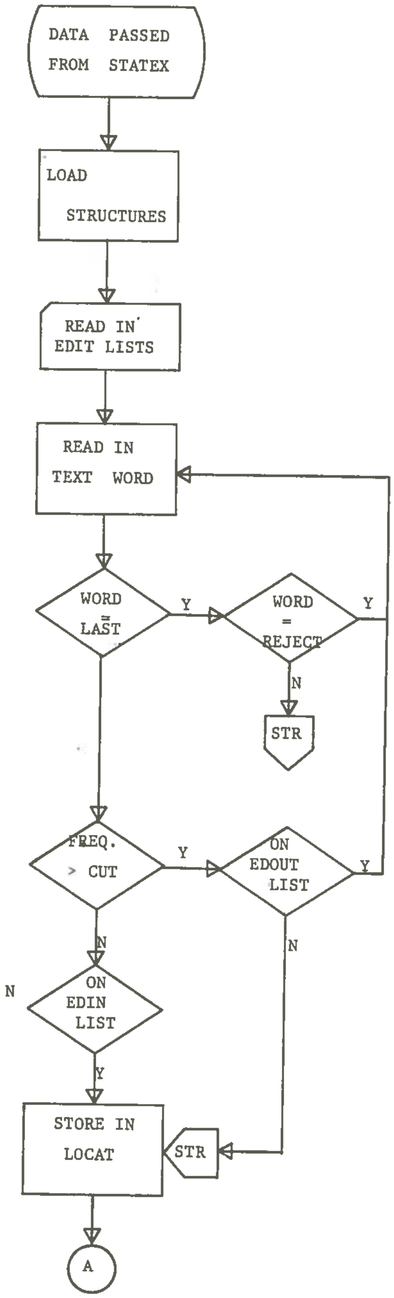
CONTEXT Programs

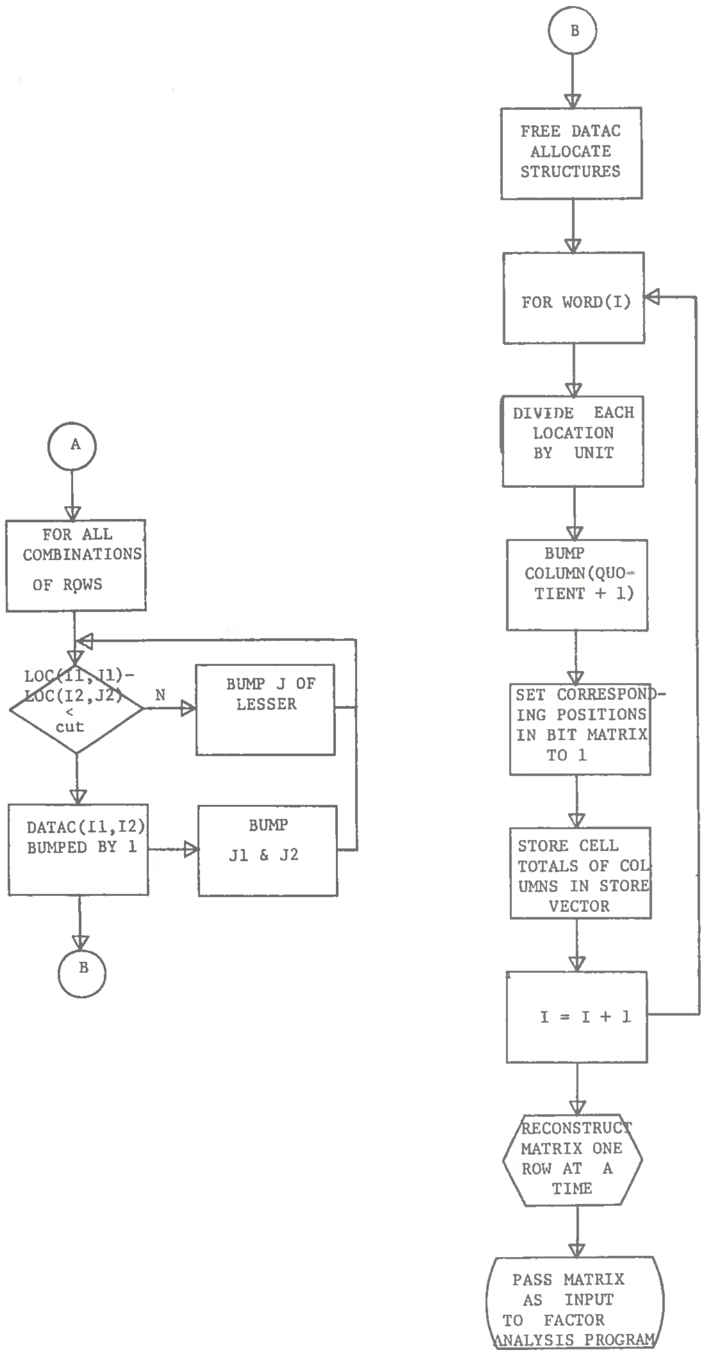
by

John B. Smith


INPUT TEXT
LINDEX
AB SORT
PREFIX
AB SORT
SUFFIX
THESAURUS PROGRAM (OPTIONAL)
STATEX
PRETEXT
PASSED DATA SET
FACTOR ANALYSIS
LINEAR SPATIAL DISPLAY
PRINTED OUTPUT

PRETEXT


DATA PASSED FROM STATEX
LOAD STRUCTURES
READ IN EDIT LISTS
READ IN TEXT WORD
WORD = LAST
Y
WORD = REJECT
Y
N
STR
FREQ. > CUT
Y
ON EDOUT LIST
Y
N
N
N
ON EDIN LIST
Y
STORE IN LOCAT
STR
A


A
FOR ALL COMBINATIONS OF ROWS
LOC(I1,J1)-LOC(I2,J2) < cut
N
BUMP J OF LESSER
DATAC(I1,I2) BUMPED BY 1
BUMP J1 & J2
B
B
FREE DATAC ALLOCATE STRUCTURES
FOR WORD(I)
DIVIDE EACH LOCATION BY UNIT
BUMP COLUMN(QUOTIENT + 1)
SET CORRESPONDING POSITIONS IN BIT MATRIX TO 1
STORE CELL TOTALS OF COLUMNS IN STORE VECTOR
I = I + 1
RECONSTRUCT MATRIX ONE ROW AT A TIME
PASS MATRIX AS INPUT TO FACTOR ANALYSIS PROGRAM

```
        PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                                        PAGE      2

STMT LEVEL NEST
  1                     PRETEXT:  PROCEDURE OPTIONS(MAIN);
                                /* PRETEXT IS THE PRINCIPLE PROGRAM IN THE CONTEXT SEQUENCE. */
                                /* IT RECEIVES AS INPUT THE TEXT DATA SET IN MATCHCOUNT,      */
                                /* WORD, AND LINEAR ORDER.  IT ALSO RECEIVES TWO EDIT LISTS: */
                                /* ONE TO EDIT IN WORDS REGARDLESS OF FREQUENCY, ONE FOR      */
                                /* EDITING OUT REGARDLESS OF FREQUENCY.
                                /* AN INPUT RECORD IS READ IN.  IF THE FREQUENCY OF THE
                                /* ROOT  GROUP IS GREATER THAN A THPRESHOLD (SET IN TERM      */
                                /* OF MEAN + N*S.D.) THE WORD IS SELECTED FOR FURTHER CON-    */
                                /* SIDERATION, PROVIDING IT HAS NOT BEEN EDITED OUT.          */
                                /* IF THE WORD IS SELECTED, IT IS LOADED  INTO A STRUCTURE    */
                                /* ALONG WITH THE LOCATIONS OF ALL TOKENS.  FROM THIS         */
                                /* STRUCTURE TWO THINGS ARE COMPUTED.  A MATRIX OF          */
                                /* CO-OCCURRENCES EXPRESSED IN ABSOLUTE TERMS IS COMPUTED.    */
                                /* THIS PROCEDURE COMPUTES THE NUMBER OF TIMES EACH PAIR      */
                                /* OF  WORDS OCCURRS WITHIN A GIVEN NUMBER OF WORDS OF EACH   */
                                /* OTHER.  AFTER THIS, THE PROGRAM COMPUTES A DATA MATRIX     */
                                /* THAT IS FED TO THE 'CANNED' FACTOR ANALYSIS PROGRAM.       */
                                /* THIS MATRIX CONSISTS OF THE NUMBER  OF OCCURRENCES OF A    */
                                /* GIVEN TERM PER GRID-UNIT OF TEXT.  FROM THIS THE FACTOR    */
                                /* ANALYSIS PROGRAM DEVELOPS FACTORS OF  WORD CLUSTERS.       */

                                /* LOCAT IS THE MAIN STORAGE STRUCTURE INTO WHICH EACH WORD   */
                                /* IS LOADED ALON WITH ITS LOCATIONS WITHIN THE TEXT.         */
                                /* THE SIXE PARAMETER FOR THIS MATRIX IS PASSED FROM STATEX   */
  2      1              DCL    01 LOCAT(NMAT) CONTROLLED,
                               02 WORD CHAR(5),
                               02 MATCH FIXED DEC(5),
                              02 NLOC FIXED DEC(3),
                               02 LOC (MAX) FIXED DEC(6);
  3      1              DCL TEMP FIXED DEC(5) INITIAL(0);

                                /* DATAC HOLDS THE MATRIX OF ABSOLUTE CO-OCCURRENCE.          */
  4      1              DCL DATAC(NMAT,NMAT) FIXED DEC(3) CONTROLLED;

  5      1              DCL LIN FIXED DEC(5) INITIAL(0),
                          COUNT FIXED DEC(2) INITIAL(0),
                           MAT FIXED DEC(5) INITIAL(0),
                           FREQ FIXED DEC(4) INITIAL(0),
                          LSTMAT FIXED DEC(5) INITIAL(0),
                           X FIXED DEC(3) INITIAL(1),
                           CH CHAR(1);
  6      1              DCL MAX FIXED DEC(3) INITIAL(0),
                            CUT FIXED DEC(3) INITIAL(0),
                            NMAT FIXED DEC(3) INITIAL(0);
  7      1              DCL WRD CHAR(6) INITIAL(' ');
  8      1              DCL ENVIR FIXED DEC(3) INITIAL(0);

                                /* ED1 AND ED2 HOLD THE TWO EDIT LISTS.                       */
  9      1              DCL ED1 (100) FIXED DEC(6) INITIAL(0);
 10      1              DCL ED2 (100) FIXED DEC(6) INITIAL(0);

 11      1              DCL X1 FIXED DEC(3) INITIAL(1);
 12      1              DCL X2 FIXED DEC(3) INITIAL(1);
 13      1              DCL T1 FIXED DEC(1) INITIAL(0);
 14      1              DCL T2 FIXED DEC(1) INITIAL(0);
```

```
        PRETEXT:   PROCEDURE OPTIONS(MAIN);                                                                   PAGE    3

STMT LEVEL NEST

 15     1          DCL REJECT FIXED DEC(6) INITIAL(0);
 16     1          DCL CLOSE1 FIXED DEC(1) INITIAL(0);
 17     1          DCL CLOSE2 FIXED DEC(1) INITIAL(0);
 18     1          DCL CT2 FIXED DEC(2) INITIAL(0);
 19     1          DCL NAMED12 FIXED DEC(2) INITIAL(0);
 20     1          DCL NAMELEFT FIXED DEC(3) INITIAL(0);
 21     1          DCL BLANK CHAR(1) INITIAL(' ');
 22     1          DCL NUMB FIXED DEC(2) INITIAL(0);
 23     1          DCL CT1 FIXED DEC(3) INITIAL(0);
 24     1          ON ENDFILE(STAT) GO TO OUT1;

                           /* CUT, MAX, AND NMAT ARE RECEIVED FROM STATEX AS PARAMETERS.*/
 26     1          GET FILE(SCRAT) EDIT(CUT,CH) (F(3), X(76), A(1));
 27     1          GET FILE(SCRAT) EDIT(NMAT,CH) (F(3), X(75), A(1));
 28     1          GET FILE(SCRAT) EDIT(MAX,CH) (F(3), X(76), A(1));

                           /* ENVIR IS THE USER-SPECIFIED PARAMETER FOR DETERMINING     */
                           /* CO-OCCURRENCE.                                            */
 29     1          ENVIR = 40;

 30     1          PUT EDIT(CUT)(SKIP, F(3));
 31     1          PUT EDIT(NMAT)(SKIP, F(3));
 32     1          PUT EDIT(MAX )(SKIP, F(3));

                           /* STRUCTURES AND VARIABLES ARE ALLOCATED AND INITIALIZED    */
 33     1          ALLOCATE DATAC(NMAT,NMAT);
 34     1          ALLOCATE LOCAT;
 35     1          WORD(*) = ' ';
 36     1          MATCH(*) = 0;
 37     1          NLOC(*) = 0;
 38     1          LOC(*,*) = 0;
 39     1          DATAC(*,*) = 0;
 40     1          ED1(*) = 0;
 41     1          ED2(*) = 0;


                           /* EDIN IS THE SUBPROCEDURE FOR EDITING IN WORDS REGARDLESS  */
                           /* OF FREQUENCY                                              */
 42     1          EDIN:  PROCEDURE;
 43     2          T1 = 0;
 44     2          IF CLOSE1 = 1 THEN GO TO ENDIN;
 46     2          B:  IF MAT = ED1(X1)
 47     2          THEN DO;
 48     2          T1 = 1;
 49     2          MATCH(I + 1) = MAT;
 50     2          WORD(I + 1) = WRD;
 51     2          LSTMAT = MAT;
 52     2          NLOC(I + 1) = NLOC(I + 1) + 1;
 53     2          LOC((I + 1),1) = LIN;
 54     2          GO TO ENDIN;
 55     2          END;
 56     2          IF MAT < ED1(K1) THEN GO TO ENDIN;
 58     2          ELSE X1 = X1 + 1;
 59     2          IF ED1(K1) = 0 THEN DO;
```

```
        PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                              PAGE      4

 STMT LEVEL NEST

  61      2            CLOSE1 = 1;
  62      2            GO TO  ENDIN;
  63      2            END;
  64      2            IF X1 > 100 THEN DO;
  66      2            CLOSE1 = 1;
  67      2            GO TO ENDIN;
  68      2            END;
  69      2            ELSE GO TO B;
  70      2            ENDIN:   END EDIN;


                               /* SIMIALRLY,  EDIUT REJECTS WORDS REGARDLESS OF FREQUENCY   */
  71      1            EDOUT:  PROCEDURE;
  72      2            T2 = 0;
  73      2            IF CLOSE2 = 1 THEN GO TO ENDOUT;
  75      2            C:  IF MAT = ED2(X2)
  76      2            THEN DO;
  77      2            REJECT = MAT;
  78      2            T2 = 1;
  79      2            GO TO ENDOUT;
  80      2            END;
  81      2            IF MAT < ED2(X2) THEN GO TO ENDOUT;
  83      2            ELSE X2 = X2 + 1;
  84      2            IF ED2(X2) = 0 THEN DO;
  86      2            CLOSE2 = 1;
  87      2            GO TO ENDOUT;
  88      2            END;
  89      2            IF X2 > 100 THEN DO;
  91      2            CLOSE2 = 1;
  92      2            GO TO ENDOUT;
  93      2            END;
  94      2            ELSE GO TO C;
  95      2            ENDOUT:  END EDOUT;


                              /* THE EDIN AND EDOUT LISTS ARE READ IN                              */
  96      1            DO I = 1 TO 100;
  97      1    1       ON ENDFILE(SYSIN) GO TO OUT4;
  99      1    1       GET FILE(SYSIN) EDIT(ED1(I), CH)(F(5), X(73), A(1));
 100      1    1       IF ED1(I) = 999999
 101      1    1       THEN DO;
 102      1    1       ED1(I) = 0;
 103      1    1       GO TO OUT3;
 104      1    1       END;
 105      1    1       END;
 106      1            OUT3:  DO I = 1 TO 100;
 107      1    1       GET FILE(SYSIN) EDIT(ED2(I), CH)(F(5), X(73), A(1));
 108      1    1       END;
 109      1            OUT4:

                       DO I = 1 TO 100;
 110      1    1       PUT EDIT(ED1(I))(SKIP, F(6));
 111      1    1       END;
 112      1            DO I = 1 TO 100;
```

```
         PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                              PAGE      5

STMT LEVEL NEST

113     1    1    PUT EDIT(ED2(I))(SKIP, F(6));
114     1    1    END;

115     1         DO I = 1 TO NMAT;
116     1    1    X = 2;
117     1    1    A:
                          /* A TEXT WORD IS READ IN                                                  */
                  GET FILE(STAT) EDIT(LIN, MAT, FREQ, WRD, CH)
                  (X(2), F(6), X(3), F(5), F(4), A(6), X(11), A(1));

                          /* AFTER THE INITIAL PROCESSING OF A WORD TYPE, ALL                        */
                          /* SUBSEQUENT TOKENS ARE PROCESSED SIMILARLY.                              */
118     1    1    IF MAT = REJECT THEN GO TO A;

120     1    1    IF MAT = LSTMAT THEN DO;
122     1    1    LOC(I,X) = LIN;   /* LOCATION IS LOADED INTO STRUCTURE                             */
123     1    1    NLOC(I) = NLOC(I) + 1;
124     1    1    X = X + 1;
125     1    1    GO TO A;
126     1    1    END;

127     1    1    IF LSTMAT = 0 THEN DO;
                          /* IF FREQ. IS GREATER THAN THRESHOLD, CHECK EDIT OUT LIST                 */
129     1    1    IF FREQ >= CUT THEN DO;
131     1    1    CALL EDOUT;
132     1    1    IF T2 = 1 THEN GO TO A;
                          /* IF WORD IS NOT EDITTED OUT, LOAD DATA INTO STRUCTURE                    */
134     1    1    MATCH(1) = MAT;
135     1    1    WORD(1) = WRD;
136     1    1    LSTMAT = MAT;   /* PROVIDES CHECK FOR SUBSEQUENT RECORDS                         */
137     1    1    NLOC(I) = NLOC(I) + 1;
138     1    1    LOC(I,1) = LIN;
139     1    1    END;
140     1    1    GO TO A;
141     1    1    END;

142     1    1    IF FREQ >= CUT THEN DO;
144     1    1    CALL EDOUT;
145     1    1    IF T2 = 1 THEN GO TO A;
147     1    1    MATCH(I + 1) = MAT;
148     1    1    WORD(I + 1) = WRD;
149     1    1    LSTMAT = MAT;
150     1    1    NLOC(I + 1) = NLOC(I + 1) + 1;
151     1    1    LOC((I + 1),1) = LIN;
152     1    1    GO TO ENDI;
153     1    1    END;

                          /* IF FREQ. IS LESS THAN THRESHOLD, CHECK EDIT IN LIST                     */
154     1    1    CALL EDIN;
155     1    1    IF T1 = 1 THEN GO TO ENDI;
157     1    1    GO TO A;

158     1    1    ENDI:  END;
```

```
       PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                                     PAGE    6

STMT LEVEL NEST

159    1          OUT1:
                  PUT PAGE;

                          /* AT THIS POINT ALL OF THE INPUT RECORDS HAVE BEEN CHECKED  */
                          /* AND THE STRUCTURES OF SELECTED WORDS COMPILED.           . */
160    1          DO I = 1 TO NMAT;
161    1     1    IF MATCH(I) = 0 THEN GO TO OUT2;
163    1     1    END;
164    1          OUT2:  NMAT = I - 1;

                          /* EACH VECTOR OF LOCATIONS IS CHECKED FOR SEQUENTIAL ORDER  */
165    1          DO I = 1 TO NMAT;
166    1     1    DO J = 1 TO (NLOC(I) - 1);
167    1     2    IF LOC(I,J) > LOC(I,J + 1) THEN DO;
169    1     2    TEMP = LOC(I,J);
170    1     2    LOC(I,J) = LOC(I,J +1);
171    1     2    LOC(I,J + 1) = TEMP;
172    1     2    DO J2 = J TO 2 BY -1 WHILE(LOC(I,J2) < LOC(I,J2-1));
173    1     3    IF LOC(I,J2) < LOC(I,J2 -1) THEN DO;
175    1     3    TEMP = LOC(I,J2);
176    1     3    LOC(I,J2) = LOC(I,J2-1);
177    1     3    LOC(I,J2-1) = TEMP;
178    1     3    END;
179    1     3    END;
180    1     2    END;
181    1     2    END;
182    1     1    END;

                          /* THE MAIN DATA STRUCTURE IS PRINTED FOR MANUAL REFERENCE  */
183    1          DO K = 1 TO NMAT;
184    1     1    PUT SKIP(2);
185    1     1    PUT EDIT(LOCAT(K))(A(6), X(2), F(5), X(2), F(3), SKIP(1), (MAX)
                  (X(2), F(5)));
186    1     1    END;

                          /* THE MATRIX OF CO-OCCURRENCES EXPRESSED IN ABSOLUTE TERMS  */
                          /* IS COMPUTED IN THE NEXT BLOCK OF CODE.  THE PROCEDURE     */
                          /* WORKS BY 'GRABBING' OUT EACH PAIR OF LOCATION VECTORS     */
                          /* CHECKING PAIRS OF LOCATIONS FOR A DIFFERENCE IN VALUE LESS*/
                          /* THAN THE SPECIFIED ENVIRONMENT PARAMETER.                  */
187    1          DO I1 = 1 TO NMAT;
188    1     1    DO I2 = 1 TO NMAT;
189    1     2    TEST1:  IF I1 = I2
190    1     2    THEN DO;
191    1     2    COUNT = NLOC(I1);
192    1     2    GO TO I2END;
193    1     2    END;
194    1     2    COUNT = 0;
195    1     2     J2 = 1;
196    1     2    DO J1 =1 TO MAX WHILE(LOC(I1,J1) ¬= 0);
197    1     3    TEST2:  IF J2 >MAX THEN GO TO I2END;
199    1     3    IF LOC(I2,J2) = 0 THEN GO TO I2END;
```

```
STMT LEVEL NEST

201     1    3    IF ABS(LOC(I1,J1) - LOC(I2,J2)) <= ENVIR
202     1    3    THEN DO;
203     1    3    COUNT = COUNT + 1;
204     1    3    J2 = J2 + 1;
205     1    3    GO TO J1END;
206     1    3    END;
207     1    3    IF LOC(I1,J1) < LOC(I2,J2) THEN GO TO J1END;
209     1    3    ELSE DO;
210     1    3    J2 = J2 + 1;
211     1    3    GO TO TEST2;
212     1    3    END;
213     1    3    J1END:  END;
214     1    2    I2END:  DATAC(I1,I2) = COUNT;
215     1    2    DATAC(I2,I1) = COUNT;
216     1    2    END;
217     1    1    I1END:  END;

                          /* THE TABLE OF CO-OCCURRENCES IS PRINTED OUT                    */
218     1         PUT PAGE;
219     1         DO K = 1 TO NMAT;
220     1    1    PUT SKIP(2);
221     1    1    PUT EDIT(DATAC(K,*)) ((NMAT) (F(3), X(2)));
222     1    1    END;

223     1         FREE DATAC;


                          /* THE REMAINDER OF THE PROGRAM IS CONCERNED WITH THE            */
                          /* COMPUTATION OF THE INPUT DATA FOR THE FACTOR ANALYSIS          */
                          /* PROCEDURE.  BECAUSE OF STORAGE CONSTRAINTS THIS MATRIX         */
                          /* CANNOT BE COMPUTED DIRECTLY.  IT IS CONSTRUCTED A COLUMN       */
                          /* AT A TIME.  SINCE IT IS A SPARCE MATRIX, ACTUAL VALUES         */
                          /* ARE STORED IN A SINGLE VECTOR OF LOCATIONS.  POSITIONS IN      */
                          /* THE MATRIX TO BE PASSED ARE MARKED IN A BIT MATRIX OF THE      */
                          /* SAME DIMENSIONS AS THE DATA MATRIX.  WHERE THERE IS A NON      */
                          /* ZERO ENTRY, THE CORRESPONDING BIT  IS SET TO 1;  OTHERWISE*/
                          /* TO 0.  WHEN ALL COLUMNS HAVE BEEN SO COMPUTED THE MATRIX       */
                          /* TO BE PASSED IS RECONSTRUCTED A ROW AT A TIME BY               */
                          /* EXTRACTING THE NON-ZERO ELEMENTS FROM THE STORAGE VECTOR       */
                          /* AND INSERTING THEM IN THEIR PROPER POSITONS .                  */

224     1         DCL BIS(NUNIT,NMAT) BIT(1) PACKED CONTROLLED;
225     1         DCL FTOT FIXED DEC(6) INITIAL(0);
226     1         DCL COL(NUNIT) FIXED DEC(2) CONTROLLED;
227     1         DCL COLTOT(NMAT) FIXED DEC(4) CONTROLLED;
228     1         DCL UNIT FIXED DEC(4) INITIAL(100);
229     1         UNIT = 50;
230     1         DCL NUNIT FIXED DEC(4) INITIAL(100);
231     1         DCL LFIELD(FTOT) FIXED DEC(2) CONTROLLED;
232     1         DCL X3 FIXED DEC(5) INITIAL(1);
233     1         DCL ROW (NMAT) FIXED DEC(2) CONTROLLED;
234     1         DCL POSIT FIXED DEC(5) INITIAL(0);
```

```
        PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                   PAGE     8

 STMT LEVEL NEST

  235     1          NUNIT = 19967/UNIT + 1;
  236     1          PUT PAGE;  PUT SKIP;  PUT DATA(NUNIT);
  239     1          DO I = 1 TO NMAT;
  240     1     1    FTOT = FTOT + NLOC(I);
  241     1     1    END;
  242     1          PUT SKIP;  PUT DATA(FTOT);

  244     1          ALLOCATE BIG;
  245     1          ALLOCATE COL(NUNIT);
  246     1          ALLOCATE ROW(NMAT);
  247     1          ALLOCATE COLTOT(NMAT);
  248     1          ALLOCATE LFIELD(FTOT);
  249     1          BIG(*,*) = 0;
  250     1          ROW(*) = 0;
  251     1           COLTOT(*) = 0;

  252     1          DO J = 1 TO NMAT;
  253     1     1    COL(*) = 0;

  254     1     1    DO I = 1 TO MAX;
  255     1     2    IF LOC(J,I) = C
  256     1     2    THEN IF I > 1      /* JUST FOR PRASSER, CHAPT. U*/
  257     1     2    THEN GO TO OUT5;
  258     1     2    COL((LOC(J,I)/UNIT) + 1) = COL((LOC(J,I)/UNIT) + 1) + 1;
  259     1     2    END;

  260     1     1    OUT5:  DO I4 = 1 TO NUNIT;
  261     1     2    IF COL(I4) ¬= C THEN DO:
  263     1     2    BIG(I4,J) = '1'B;
  264     1     2    PUT SKIP;  PUT DATA(BIG(I4,J));
  266     1     2    LFIELD(X3) = COL(I4);
  267     1     2    PUT DATA(LFIELD(X3));
  268     1     2    X3 = X3 + 1;         ,
  269     1     2    COLTOT(J) = COLTOT(J) + 1;
  270     1     2    END;
  271     1     2    END;

  272     1     1    END;


  273     1          DCL CT FIXED DEC(3) INITIAL(0);
  274     1          DO I = 1 TO 15 WHILE(CT <= 0);
  275     1     1    CT = 80*I - NMAT*2;
  276     1     1    END;
  277     1          CT = 80 - CT;
  278     1          CT2 = CT/2;
  279     1          NUMB = I - 2;
  280     1          PUT SKIP;  PUT DATA(NMAT, NUMB, CT2);
  282     1          NAMED12 = NMAT/12;
  283     1          NAMELEFT = NMAT - 12*NAMED12;
  284     1          PUT FILE(FATPASS) EDIT('CONTEXT:  FACTOR ANALYSIS OF CHAPT. 1 OF PORTRA
                     IT--UNIT= ',UNIT, BLANK)(A, F(4), X(18), A(1));
  285     1          PUT FILE(FATPASS) EDIT(NMAT,NUNIT, '0', '(', NUMB, '(40(F2.0)/),',
       PRETEXT:  PROCEDURE OPTIONS(MAIN);                                                              PAGE     5

STMT LEVEL NEST

                         CT2,  '(F2.0))', BLANK)
                         (F(5), F(5), X(21), A(1), X(4), A, F(2), A  F(2), A, X(19), A(1));
 286     1               DO I = 1 TO NAMED12;
 287     1     1         DO J = 1 TO 12;
 288     1     2         PUT FILE(PATPASS) EDIT(WORD(12*(I - 1) + J))(A(5));
 289     1     2         END;
 290     1     1         PUT FILE(PATPASS) EDIT('        ')(A(8));
 291     1     1         END;
 292     1               DO I = (12*NAMED12 + 1) TO NMAT;
 293     1     1         PUT FILE(PATPASS) EDIT(WORD(I))(A(6));
 294     1     1         CT1 = CT1 + 1;
 295     1     1         END;
 296     1               PUT FILE(PATPASS) EDIT((120) ' ')(A(80 - CT1*6));

 297     1               PUT PAGE;
 298     1               DO J = 1 TO NUNIT;
 299     1     1         ROW(*) = 0;
 300     1     1         DO I = 1 TO NMAT;
 301     1     2         IF BIG(J,I) = 1
 302     1     2         THEN DO;
 303     1     2         POSIT = 0;
 304     1     2         DO I3 = 1 TO (I - 1) WHILE((I-1)>= 1);
 305     1     3         POSIT = POSIT + COLTOT(I3);
 306     1     3         END;
 307     1     2         DO J3 = 1 TO J;
 308     1     3         IF BIG(J3,I) = 1 THEN POSIT = POSIT + 1;
 310     1     3         END;
 311     1     2         ROW(I) = LFIELD(POSIT);
 312     1     2         PUT SKIP;  PUT DATA(POSIT, LFIELD(POSIT));
 314     1     2         END;
 315     1     2         END;
 316     1     1         PUT FILE(PATPASS) EDIT(ROW(*))((NMAT) F(2));
 317     1     1         PUT FILE(PATPASS) EDIT((120) ' ')(A(80 - CT));
 318     1     1         PUT EDIT(ROW(*))(SKIP, (NMAT) (X(2), F(2)));
 319     1     1         END;

 320     1               END PRETEXT;
```

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)*

University of North Carolina
Chapel Hill, N. C.

2a. REPORT SECURITY CLASSIFICATION

Unclassified

2b. GROUP

3. REPORT TITLE

Automated Language Analysis, 1968-1969: Report on Research for the Period March 1, 1968 - February 28, 1969.

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Sedelow, Sally Yeates

6. REPORT DATE

1 March, 1969

7a. TOTAL NO. OF PAGES

7b. NO. OF REFS

8a. CONTRACT OR GRANT NO.

N00014-67-A-0321-0-0001

b. PROJECT NO. Task No. NR 348-005

c. Office of Naval Research, U. S. Navy

d.

9a. ORIGINATOR'S REPORT NUMBER(S)

9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

Distribution of this document is unlimited.

11. SUPPLEMENTARY NOTES

12. SPONSORING MILITARY ACTIVITY

13. ABSTRACT

This report describes research directed toward unequivocally characterizing the language usage of any writer or speaker. During the past year a PREFIX program has increased the root-grouping capability of the VIA package, which is a set of programs designed for content, or thematic, analysis of a written or spoken language unit. A ring-structure version of VIA which provides great search flexibility as well as the potential for extended explorations of semantic relationships, as revealed by interconnecting rings, has been further developed. Research on the nature of thesauri has continued and Roget's International Thesaurus has been keypunched to facilitate computer-aided research on its structure. A set of programs designed to show co-occurrence patterns has been implemented, as have procedures for produdimg non-verbal representations of language-usage patterns.

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Stylistic analysis						
Automated language analysis						
Thesaurus						
Content Analysis						